UNIVERSIDAD POLITÉCNICA DE MADRID

FACULTAD DE INFORMÁTICA

TESIS DE MÁSTER
MÁSTER UNIVERSITARIO EN SOFTWARE Y SISTEMAS

ANÁLISIS DE CARACTERÍSTICAS ESTÁTICAS DE FICHEROS EJECUTABLES PARA LA CLASIFICACIÓN DE MALWARE

Un trabajo presentado en cumplimiento de requisitos para el grado de Máster Universitario en Software y Sistemas

Autor: Richard Rivera Guevara

Director: PhD Juan Caballero

MADRID - JUNIO DEL 2014

Para mi persona favorita en el mundo Jennifer

Richard Rivera

AGRADECIMIENTOS

Un especial reconocimiento a Institute IMDEA Software por la oportunidad de realizar este trabajo junto a reconocidos científicos como el Dr. Juan Caballero. Siempre fue uno de mis anhelos ser un científico y en este corto tiempo me he podido sentir así, como un pequeño científico que da sus primeros pasos guiado por expertos.

Gracias a la Universidad Politécnica de Madrid por permitirme adquirir tan diversos conocimientos.

Gracias a la Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación del Ecuador por otorgarme esta beca de estudios y al presidente de la República del Ecuador Rafael Correa que con su gestión he podido ser parte de la revolución del conocimiento de Ecuador.

A Ricardo, Nelly, Estefany y Emily sin su apoyo todos estos logros realizados en tan poco tiempo en esta legendaria travesía hubieran sido casi imposibles.

A los amigos y familiares que en la distancia me apoyaron constantemente, a Beatriz y Fernando por los ánimos y apoyo en mi primera etapa universitaria, con sus charlas fue donde empezó este sueño que va haciéndose realidad poco a poco.

Y por último a mi madre que estando tan lejos me hacía sentir su apoyo todos los días.

RESUMEN

El Malware es una grave amenaza para la seguridad de los sistemas. Con el uso generalizado de la World Wide Web, ha habido un enorme aumento en los ataques de virus, haciendo que la seguridad informática sea esencial para todas las computadoras y se expandan las áreas de investigación sobre los nuevos incidentes que se generan, siendo una de éstas la clasificación del malware. Los "desarrolladores de malware" utilizan nuevas técnicas para generar malware polimórfico reutilizando los malware existentes, por lo cual es necesario agruparlos en familias para estudiar sus características y poder detectar nuevas variantes de los mismos.

Este trabajo, además de presentar un detallado estado de la cuestión de la clasificación del malware de ficheros ejecutables PE, presenta un enfoque en el que se mejora el índice de la clasificación de la base de datos de Malware MALICIA utilizando las características estáticas de ficheros ejecutables Imphash y Pehash, utilizando dichas características se realiza un clustering con el algoritmo clustering agresivo el cual se cambia con la clasificación actual mediante el algoritmo de majority voting y la característica icon_label, obteniendo un Precision de 99,15% y un Recall de 99,32% mejorando la clasificación de MALICIA con un F-measure de 99,23%.

ABSTRACT

Malware is a serious threat to the security of systems. With the widespread use of the World Wide Web, there has been a huge increase in virus attacks, making the computer security essential for all computers. Near areas of research have append in this area including classifying malware into families, Malware developers use polymorphism to generate new variants of existing malware. Thus it is crucial to group variants of the same family, to study their characteristics and to detect new variants.

This work, in addition to presenting a detailed analysis of the problem of classifying malware PE executable files, presents an approach in which the classification in the Malware database MALICIA is improved by using static characteristics of executable files, namely Imphash and Pehash. Both features are evaluated through clustering real malware with family labels with aggressive clustering algorithm and combining this with the current classification by Majority voting algorithm, obtaining a Precision of 99.15% and a Recall of 99.32%, improving the classification of MALICIA with an F-measure of 99,23%.

Tabla de Contenido

Lista de tablas

Lista de Figuras

1Introducción

La clasificación del malware es un reto para los investigadores que estudian la seguridad informática. Los atacantes se valen de nuevas técnicas, sin dejar de lado las tradicionales para mejorar las técnicas de desarrollo de malware tratando de reutilizar los malware existentes. El cibercrimen organizado vende los malware y empaquetadores de estos para que usuarios comunes los usen, incrementando el acceso al malware, por lo que los investigadores tratan de descubrir nuevos modelos que permitan identificar al software malicioso, identificar las variantes de malware polimórfico para clasificarlos en familias y facilitar la detección de las nuevas variantes del mismo malware.

Este trabajo pretende presentar un enfoque que sirva de apoyo para los análisis dinámicos de malware, utilizando características estáticas, para mejorar los índices de clasificación previamente obtenidos, También se espera que se utilice este enfoque para detectar incongruencias en clasificaciones de malware realizadas previamente.

El desarrollo de este proyecto se realiza partiendo del proyecto Malicia[1] realizado por el instituto IMDEA Software, proponiendo una mejora en la clasificación actual de su base de datos de Malware MALICIA, la cual ha sido realizada previamente en dos publicaciones (Rafique & Caballero, 2013), (Nappa, Rafique, & Caballero, 2013).

Esta investigación cuenta con dos objetivos principales. Primero seleccionar características estáticas que permitan realizar la clasificación del malware. Segundo probar varios Clustering para tratar de mejorar el índice de clasificación de la base de datos de los ficheros ejecutables de MALICIA.

A continuación de esta breve introducción en el capítulo 2 se presenta el estado de la cuestión en el cual se da un repaso general sobre los ficheros ejecutables PE, mencionando la composición del formato PE las secciones que se pueden encontrar en éste. También se presentan las definiciones y características principales de los más comunes tipos de malware con los que nos podemos encontrar, así como los tipos de análisis que se pueden realizar. La parte final de este capítulo presenta los antecedentes relacionados con los objetivos de este trabajo presentando los inconvenientes con los

[1] http://malicia-project.com/

que se enfrenta la clasificación de malware y algunas de las técnicas de clasificación que existen para contrarrestar estos inconvenientes.

El capítulo 3 expone el actual problema por el que surge la necesidad de realizar este trabajo. En él se explican detalladamente los aspectos de la base de datos de malware MALICIA y se expone el problema de utilizar características estáticas para la clasificación de malware, las soluciones propuestas a estos problemas se presentan en el siguiente capítulo.

En el capítulo 4 se plantea el método utilizado para cumplir los objetivos. En él se explica las características estáticas Imphash y Pehash que serán las que a lo largo de este capítulo participaran en distintos clustering para realzar una nueva clasificación de MALICIA. Para calcular la validez de estas clasificaciones se utilizará las métricas RS, JC, FM; Precision, Recall y F-Measure. Luego de presentar las métricas se procede a explicar el proceso para obtener los resultados y los inconvenientes que se superaron. En el capítulo 5 se presenta un análisis consolidado sobre el proceso de las soluciones adoptadas y se culmina con las conclusiones y trabajos futuros que pueden realizarse a partir de este trabajo.

2Estado de la cuestión

Este capítulo introduce los principales temas relacionados con, entre otros los ficheros ejecutables y sus características, el malware, su clasificación y las técnicas para su análisis.

2.1 Ficheros ejecutables

Un archivo o fichero ejecutable, es un fichero binario cuyo contenido se interpreta por la computadora como un programa. Generalmente, contiene instrucciones en código máquina de un procesador en concreto, pero también puede contener bytecode que requiera un intérprete para ejecutarlo. Además suele contener llamadas a funciones o librerías específicas de un sistema operativo (llamadas al sistema). Sin embargo en un sentido más general, un programa ejecutable no tiene por qué necesariamente contener código de máquina, sino que puede tener instrucciones a interpretar por otro programa. Este tipo de ejecutables son conocidos con el nombre de scripts.

En la mayoría de los sistemas modernos, un archivo ejecutable contiene mucha información que no es parte del programa en sí: recursos como textos e imágenes, requisitos del entorno de ejecución, información simbólica y de depuración, u otra información que ayude al sistema operativo a ejecutar el programa.

Los ficheros ejecutables generalmente tienen un formato específico de acuerdo al sistema operativo o familias de sistemas operativos para el que fue diseñado, esto es debido a que cada sistema utiliza una arquitectura diferente para realizar las llamadas al sistema.

Entre los formatos que figuran, los de uso más común son PE (en Microsoft Windows), ELF (en Linux y la mayoría de las otras versiones de Unix), Mach-O (en OS X y iOS) y MZ (en DOS). Esta tesis se centra en el formato PE de Microsoft.

2.1.1 Portable Ejecutable (PE)

El formato Portable Ejecutable (PE) es un formato de archivo para archivos ejecutables, de código objeto, bibliotecas de enlace dinámico (DLL), archivos de fuentes, y otros archivos usados en versiones de 32 bit y 64 bit de los sistema operativos Microsoft Windows. El término «portable» refiere a la versatilidad del formato en numerosos

ambientes de arquitecturas de software de sistema operativo. El formato PE es una estructura de datos que encapsula en secciones la información necesaria para el cargador de Windows. Esto incluye las referencias hacia las bibliotecas de enlace dinámico para el enlazado, la importación y exportación de las tablas de la API de Windows y el framework de.NET, gestión de los datos de los recursos y los datos de almacenamiento local de subprocesos. En sistemas operativos basados en Windows NT, el formato PE es usado para los tipos de archivo con extensión EXE, DLL, SYS, y otros tipos de archivo. PE es una versión modificada del formato COFF de Unix. Además, PE/COFF es un nombre alternativo en el desarrollo de Windows.

2.1.1.1 Formato de un PE

Un archivo PE consiste de una serie de cabeceras y secciones que indican al enlazador dinámico cómo asignar el archivo en memoria. Una imagen ejecutable consta de varias zonas diferentes, cada una de las cuales requieren diferente protección de memoria, por lo que el inicio de cada sección debe estar alineado a un límite de página. Por ejemplo, generalmente la sección.text (que contiene el código del programa) es mapeada como «ejecutar/sólo lectura», y la seccion.data (que contiene las variables globales) es mapeada como «no-ejecutar/lectura-escritura». Sin embargo, para evitar el desperdicio de espacio, las diferentes secciones no son alineadas en página en disco. Parte del trabajo del enlazador dinámico es mapear cada sección en memoria de manera individual y asignar los permisos correctos para las regiones resultantes, de acuerdo con las instrucciones que se encuentran en las cabeceras.

La Figura 1, ilustra el formato de un fichero Microsoft PE ejecutable (Microsoft1, 2013).

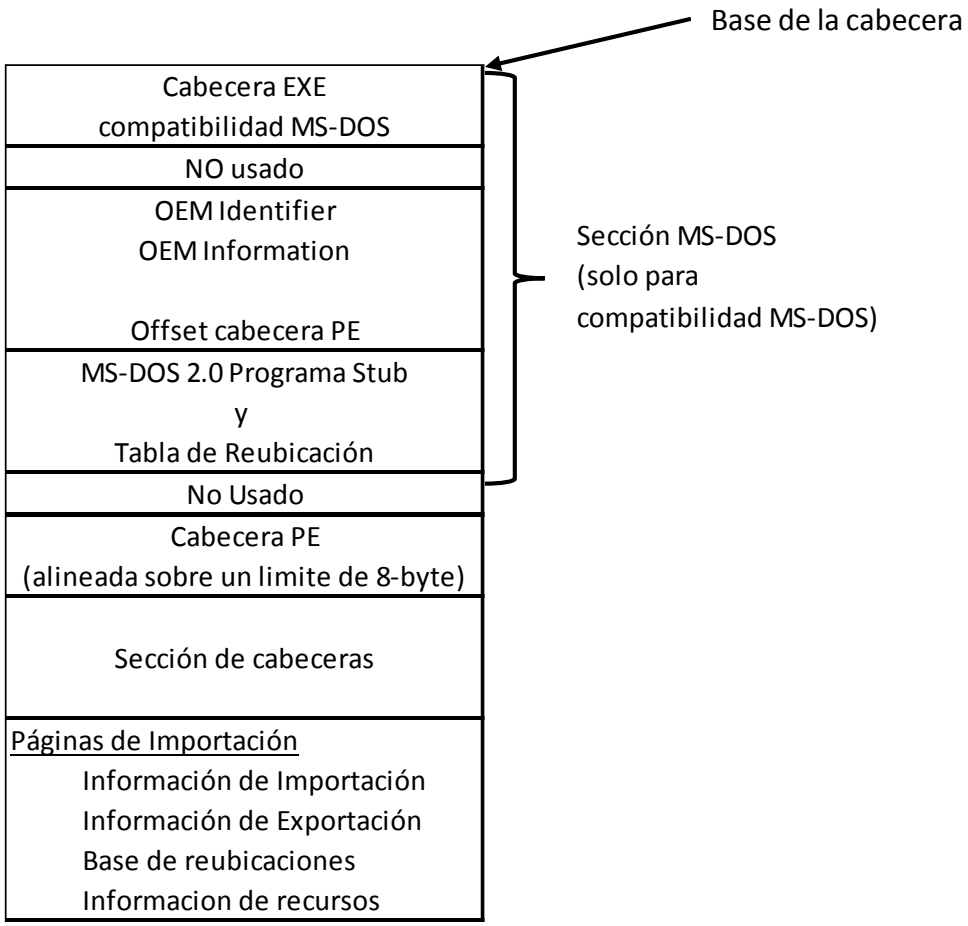

Figura 1 Típico Diseño de un fichero Portable EXE (Microsoft1, 2013)

Todas las cabeceras y secciones que se pueden ver en el formato PE presentado en la Figura 1, contienen la información que utiliza el cargador de Windows para crear una imagen (copia del fichero PE), que utilizará el enlazador dinámico para asignar esta imagen en la memoria del ordenador de acuerdo a la información que tenga el fichero en sus secciones. Toda la especificación en detalle del formato PE de Microsoft se puede encontrar en (Microsoft1, 2013) (Pietrek, 1994) (Carrera, 2005).

De acuerdo a la Figura 1 luego de la sección de MS-DOS comienza la cabecera PE, seguida de la sección de cabeceras, la cual es una tabla que hace referencia a todas las secciones que contiene el archivo, en el cuadro inferior y el último de la Figura 1 se encuentran las Páginas de importación a las cuales nos referíamos como las secciones del PE.

Para el estudio concerniente a esta tésis varios de los scripts creados utilizarán toda la estructura del formato PE, pero principalmente se enfocaran en la cabecera de PE y en la sección información de importación o tabla de importación.

2.1.1.2 Tabla de secciones de un PE

La tabla de secciones se encuentra justo detrás de la cabecera PE. Esta es una tabla que contiene varias estructuras IMAGE_SECTION_HEADER. Estas estructuras contienen información sobre las secciones de los binarios para ser cargados en la memoria. El campo NumberOfSections de la estructura IMAGE_FILE_HEADER indica cuantas

entradas hay en la tabla. El máximo soportado por Windows es de 96 secciones. De estas 96 secciones únicamente 8 son obligatorias en la mayoría de los casos (Microsoft1, 2013). Las demás contienen recursos adicionales que necesita el fichero PE dependiendo del propósito que tenga el ejecutable. En la Tabla 1 se presenta un prototipo de una de las estructuras que suelen presentarse en la tabla de secciones.

```
typedef struct _IMAGE_SECTION_HEADER {
  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
  union {
     DWORD PhysicalAddress;
     DWORD VirtualSize;
  } Misc;
  DWORD VirtualAddress;
  DWORD SizeOfRawData;
  DWORD PointerToRawData;
  DWORD PointerToRelocations;
  DWORD PointerToLinenumbers;
  WORD  NumberOfRelocations;
  WORD  NumberOfLinenumbers;
  DWORD Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;
```

Tabla 1 Ejemplo de un prototipo de una tabla de secciones

Las secciones más comunes que comúnmente se puede encontrar en un fichero PE se presentan en la Tabla 2.

Nombre	**Descripción**
.text	El código (instrucciones) del programa
.bss	Las variables no inicializadas
.reloc	La tabla de relocalizaciones
.data	Las variables inicializadas
.rsrc	Los recursos del archivo (cursores, sonidos, menús...)
.rdata	Los datos de solo lectura

.idata	La tabla de importación
.upx	Firma de una compresión UPX, propio de software UPX
.aspack	Firma de un paquete ASPACK, propio de software ASPACK
.adata	Firma de un paquete ASPACK, propio de software ASPACK

Tabla 2 Secciones de un fichero PE

El.NET Framework de Microsoft ha ampliado el formato PE con características que soportan el Common Language Runtime (CLR). Entre las características añadidas se encuentra las secciones de cabecera CLR y datos CLR. Al cargar un binario, el cargador del sistema operativo cede la ejecución a CLR mediante una referencia en la tabla IMPORT de PE/COFF. Entonces CLR carga las secciones CLR cabecera y datos.

La sección de datos de CLR contiene dos segmentos importantes, código de metadatos y código de Intermediate Language (IL) (Microsoft2, 2014):

- Los metadatos contienen información relevante para el ensamblado, incluyendo el manifiesto de ensamblado. Un manifiesto describe el ensamblado en detalle, incluyendo la identificación única (mediante un hash, número de versión, etc.), datos en componentes exportados, información de tipo extensivo (soportado por el Common Type System (CTS)), referencias externas, y una lista de archivos dentro del ensamblado. El entorno CLR hace un uso amplio de los metadatos.
- El código IL es un código abstracto e independiente del lenguaje que satisface los requisitos del Common Intermediate Languge (CIL) del CLR. El término «intermediate» (en español «intermedio») se refiere a la naturaleza compatible entre lenguajes y entre plataformas del código IL. Este lenguaje intermedio, similar a Java bytecode, permite a las plataformas y a los lenguajes soportar el CLR común. IL soporta la programación orientada a objetos (polimorfismo, herencia, tipos abstractos, etc.), excepciones, eventos, y varias estructuras de datos.

Tabla de Importación

Una sección a mencionar es la tabla de importación de direcciones (en inglés import address table o IAT), que se usa como una tabla de búsqueda cuando la aplicación llama una función en un módulo diferente. Esto puede darse en forma de importación por ordinal o importación por nombre. Debido a que un programa compilado no puede conocer la ubicación en memoria de las bibliotecas de las que depende directamente, cada vez que se hace una llamada a la API es necesario un salto indirecto. Como el enlazador dinámico carga los módulos y los une, éste escribe las direcciones actuales en las ranuras IAT, de manera que apuntan a las ubicaciones de memoria correspondientes de la biblioteca de funciones. Aunque esto agrega un salto adicional en el costo de una llamada intra-módulo que resulta en una reducción del rendimiento, ofrece una ventaja clave: se reduce al mínimo el número de páginas de memoria que necesitan ser cambiados (copy-on-write) por el cargador, ahorrando memoria y accesos al disco. Si el compilador conoce de antemano que será una llamada inter-módulo (mediante el

atributo dllimport) puede producir código más optimizado que simplemente se traduce en una llamada indirecta.

2.2 Malware

Malware es la abreviatura de “software malicioso”, y se refiere a los programas de software diseñados para dañar o realizar otras acciones no deseadas en un sistema informático. El término Malware está formado por la unión de los términos “Mal” la cual proviene del latín “malus”, que es un prefijo que significa malo o maligno y de “Ware” que hace referencia al software, y la expresión es un término general usado por los profesionales de la informática para referirse a una variedad de software hostil, intrusivo o molesto o a un código de programa.

Muchas personas han tratado de definir el malware mediante la descripción de sus características esenciales.

Ya en 1986, Fred Cohen presentó la primera definición rigurosa de un virus informático en su tesis de doctorado (Cohen, 1985). Él escribió “Un virus puede ser descrito por una secuencia de símbolos que es capaz, cuando se interpreta en un entorno adecuado (una máquina), de modificar otras secuencias de símbolos en ese entorno mediante la inclusión de una, posiblemente evolucionada, copia de sí mismo.” Aunque su tesis sólo se centra en los virus y no consideró la cuestión más general que es el malware, El 10 de noviembre de 1983, en la Universidad de Lehigh, Cohen demostró un programa similar a un virus en un sistema VAX11/750. El programa fue capaz de instalarse, o infectar a otros objetos del sistema. Este es el nacimiento de virus informático experimental.

La definición del malware varía normalmente aumentando su alcance, a la vez que avanzan los desarrollos de los sistemas informáticos, y el Internet. El software es considerado malware basado en la intención del creador, más que en cualquier característica particular. Así que desde un punto de vista práctico, la siguiente definición es popularmente aceptado: Malware es un software diseñado para infiltrarse o dañar un sistema informático sin el consentimiento informado del propietario.

2.2.1 Tipos de malware

Con el rápido desarrollo y popularidad de Internet, los dispositivos móviles y todos los procesos donde intervienen las tecnologías de la información, el malware se ha ido complicando, mejorando y evolucionando. Empezó con los primeros virus posteriormente aparecieron los gusanos y troyanos, y ahora los más recientes rootkits y backdoors entre otros. Su evolución es rápida y sus características frecuentemente compartidas entre varios tipos, para lo cual la realización de una taxonomía o clasificación del mismo no es una tarea fácil. No obstante en los siguientes párrafos se presenta los tipos más comunes y sus principales características:

2.2.1.1 Virus y Gusanos

Los más antiguos y más conocidos tipos de malware son los gusanos y los virus. Los gusanos son programas que se propagan a través de redes de área local o de Internet con objetivos maliciosos, tratando de penetrar en máquinas remotas, el lanzamiento de

copias en los equipos víctimas y se propague aún más a las nuevas máquinas. Los gusanos utilizan diferentes sistemas de red para propagarse, como correo electrónico, mensajería instantánea, intercambio de archivos (P2P), canales IRC, redes LAN y WAN.

La mayoría de los gusanos existentes se propagan como archivos de una forma u otra, incluidos los archivos adjuntos de correo electrónico, mensajes IRC y archivos accesibles a través de redes P2P. Hay un pequeño número de gusanos que no vienen en un archivo para almacenarse y se propagan como paquetes de red y penetran directamente a la memoria RAM de la máquina de la víctima, donde se ejecuta el código malicioso.

Los gusanos utilizan una variedad de exploits para penetrar en equipos de las víctimas y posteriormente, ejecutar el código, los ***exploits*** pueden ser fragmentos de software, fragmento de datos o secuencia de comandos y/o acciones, utilizada con el fin de aprovechar una vulnerabilidad de seguridad de un sistema de información para conseguir un comportamiento no deseado del mismo. Algunos ejemplos de comportamiento erróneo son: accesos no autorizados, toma de control de un sistema de cómputo, consecución privilegios no concedidos lícitamente y la consecución de ataques de denegación de servicio. El término exploit no se circunscribe a piezas de software. Por ejemplo cuando se realiza un ataque de ingeniería social, el ardid o discurso que el atacante prepara para convencer a la víctima también se considera un exploit.

Los virus son programas que se propagan como copias de sí mismos a lo largo de una sola máquina con el fin de poner en marcha y / o ejecutar código cuando el usuario responde a una acción designada, y también penetra otros recursos dentro de la máquina de la víctima. A diferencia de los gusanos, los virus no utilizan recursos de la red para penetrar en otras máquinas. Las copias de los virus pueden penetrar en otras máquinas sólo si se accede a un objeto infectado y el código se puso en marcha por un usuario en una máquina no infectada. Esto puede suceder en las siguientes maneras: el virus infecta archivos en un recurso de red que otros usuarios pueden acceder; el virus infecta medios de almacenamiento extraíbles que luego se utiliza en una máquina limpia; o el usuario adjunta un archivo infectado a un correo electrónico y lo envía a un destinatario “saludable”. Los virus pueden ser llevados por gusanos como cargas adicionales y ellos mismos pueden incluir funcionalidad de puerta trasera o troyano que destruye los datos en la máquina infectada.

Un virus requiere la intervención del usuario para propagarse, mientras que un gusano se propaga automáticamente. Debido a esta distinción, las infecciones de transmisión por correo electrónico o documentos de Microsoft Word por ejemplo, que se basan en que el destinatario debe abrir un archivo o correo electrónico para infectar el sistema, sería clasificado como un virus en lugar de un gusano.

2.2.1.2 Troyanos

El principal objetivo de este tipo de malware es introducir e instalar otras aplicaciones en el equipo infectado, para permitir su control remoto desde otros equipos.

Los troyanos no se propagan por sí mismos, y su nombre deriva del parecido en su forma de actuar con los astutos griegos de la mitología, ya que los troyanos llegan al equipo

del usuario como un programa aparentemente inofensivo, pero, en determinados casos, al ejecutarlo instalará en el equipo infectado un segundo programa; el troyano en sí. Este es un claro ejemplo de la familia de troyanos de tipo downloader.

Los efectos de los troyanos pueden ser muy peligrosos. Al igual que los virus, tienen la capacidad de eliminar ficheros o destruir la información del disco duro. Pero además pueden capturar y reenviar datos confidenciales a una dirección externa o abrir puertos de comunicaciones, permitiendo que un posible intruso controle nuestro ordenador de forma remota. También pueden realizar acciones tales como capturar todos los textos introducidos mediante el teclado o registrar las contraseñas introducidas por el usuario. Debido a esta particular característica, son muy utilizados por los ciberdelincuentes para, robar datos bancarios.

En los últimos años y gracias a la popularización de Internet esta tendencia ha cambiado y es que los ciberdelincuentes han visto en este malware la herramienta perfecta para robar datos bancarios, nombres de usuario y contraseñas, información personal, etc. Es decir, han dado pie a la creación de una nueva categoría de malware: los troyanos bancarios y el Spyware.

2.2.1.3 Rootkit

Un rootkit es un programa que permite un acceso de privilegio continuo a una computadora pero que mantiene su presencia activamente oculta al control de los administradores al corromper el funcionamiento normal del kernel del sistema operativo. El término proviene de una concatenación de la palabra inglesa root, que significa 'raíz' (nombre tradicional de la cuenta privilegiada en los sistemas operativos Unix) y de la palabra inglesa kit, que significa 'conjunto de herramientas' (en referencia a los componentes de software que implementan este programa).

Usualmente se asocia el termino rootkit con malware, que se esconde a sí mismo y a otros programas, procesos, archivos, directorios, claves de registro, y puertos que permiten al intruso mantener el acceso a una amplia variedad de sistemas operativos como pueden ser GNU/Linux, Solaris o Microsoft Windows para remotamente comandar acciones o extraer información sensible.

Típicamente, un atacante instala un rootkit en una computadora después de primero haber obtenido un acceso al nivel raíz, ya sea por haberse aprovechado de una vulnerabilidad conocida o por haber obtenido una contraseña (ya sea por crackeo de la encriptación o por ingeniería social). Una vez que el rootkit ha sido instalado, permite que el atacante disfrace la siguiente intrusión y mantenga el acceso privilegiado a la computadora por medio de rodeos a los mecanismos normales de autenticación y autorización. Pese a que los rootktis pueden servir con muchos fines, han ganado notoriedad fundamentalmente como malware, escondiendo programas que se apropian de los recursos de las computadoras o que roban contraseñas sin el conocimiento de los administradores y de los usuarios de los sistemas afectados. Los rootkits pueden estar dirigidos al firmware, al hipervisor, al núcleo o, más comúnmente, a los programas del usuario, En general, existen dos tipos de rootkits: modo usuario y modo kernel.

Rootkits modo usuario

Su principal característica es que se ejecuta con los permisos, privilegios y configuración de seguridad de un usuario dado de alta en el sistema. Debido a que están vinculados directamente a un usuario específico y suelen operar en o por debajo del kernel del sistema operativo, estos rootkits pueden ser detectados por los usuarios con privilegios de administrador.

Rootkits modo kernel

Es malware que opera dentro del sistema operativo al mismo nivel que los controladores del hardware de la tarjeta gráfica, tarjeta de red, o el ratón, lo que hace que su detección y eliminación sea altamente compleja. Uno de los inconvenientes que tiene este tipo de malware es que está implementado para una versión específica de sistema operativo, por lo que actualizaciones que modifiquen el kernel hacen que este tipo de rootkits no funcionen.

2.2.1.4 Backdoors

Una puerta trasera o backdoor, en un sistema informático es una secuencia especial dentro del código de programación, mediante la cual se pueden evitar los sistemas de seguridad de autentificación para acceder al sistema. Estas puertas pueden ser utilizadas para fines maliciosos y espionaje no siempre son un error, pueden haber sido diseñadas con la intención de tener una entrada secreta en el programa.

Las puertas traseras también pueden ser instaladas con anterioridad en un software malicioso, para permitir la entrada de atacantes. A menudo se ha sugerido que los fabricantes de ordenadores preinstalan puertas traseras en sus sistemas para proporcionar apoyo técnico a los clientes, pero esto nunca ha sido verificada de forma fiable. Los atacantes suelen utilizar puertas traseras para asegurar el acceso remoto a un equipo, al intentar permanecer oculto a la inspección ocasional. Para instalar puertas traseras, los crackers pueden utilizar caballos de Troya, gusanos, u otros métodos.

2.2.1.5 Spyware and Adware

Un spyware típico se auto instala en el sistema afectado de forma que se ejecuta cada vez que se pone en marcha el ordenador (utilizando CPU y memoria RAM, reduciendo la estabilidad del ordenador), y funciona todo el tiempo, controlando el uso que se hace de Internet y mostrando anuncios relacionados. Sin embargo, a diferencia de los virus, no se intenta replicar en otros ordenadores, por lo que funciona como un parásito.

El spyware o programa espía es un software que recopila información de un ordenador y después transmite esta información a una entidad externa sin el conocimiento o el consentimiento del propietario del ordenador. El término spyware también se utiliza más ampliamente para referirse a otros productos que no son estrictamente spyware. Estos productos llamados Adware, realizan diferentes funciones, como mostrar anuncios no solicitados (pop-up), recopilar información privada o redirigir solicitudes de páginas. De forma general adware es cualquier programa que automáticamente muestra publicidad web al usuario durante su instalación o durante su uso para generar lucro a sus autores.

2.2.1.6 Bot

"Bot" es la abreviatura de la palabra "robot", que es otro tipo de malware y es un proceso automatizado que interactúa con otros servicios de red. Un uso típico de los robots es reunir información (como los rastreadores web), o interactuar de forma automática con la mensajería instantánea (IM), Internet Relay Chat (IRC), u otras interfaces web. El software Bot permite al operador controlar de forma remota cada sistema y agruparlos para formar lo que se conoce comúnmente como botnets.

Los atacantes pueden utilizar estos bots como proxies anónimos para ocultar sus verdaderas identidades y amplificar sus ataques. Una botnet es un gran ejército de ordenadores comprometidos a través de Internet. Los atacantes pueden usar un botnet para lanzar, de forma remota, ataques de tipo de inundaciones de peticiones en contra de sus objetivos provocando la denegación del servicio. Actualmente los robots que se encuentran son híbridos de amenazas previas. Esto significa que pueden propagarse como gusanos, esconderse de detección como muchos virus, ataques al igual que muchas herramientas, y tienen un sistema de mando y control integrado. Estos también han sabido explotar las puertas traseras abiertas por gusanos y virus, lo que les permite el acceso a las redes controladas. Los bots tratan de ocultar a sí mismos tanto como les sea posible e infectar las redes sin que los sistemas los detecten.

2.2.1.7 Ransomware

Es una clase de malware que busca un beneficio económico mediante la realización de un chantaje al usuario de la máquina infectada, como la restricción de acceso al mismo coaccionándole a pagar un rescate al cibercriminal para que la restricción sea eliminada. Ejecuta normalmente los siguientes tipos de acciones:

- Cifrado de archivos del disco duro del sistema.
- Bloqueo de cuentas de usuario.
- Bloqueo de programas.
- Bloqueo administrador de tareas.
- Muestra de un mensaje como medio de persuadir al usuario a realizar el
- Pago, entre otras.

2.2.1.8 Rogueware o Scareware

Otro tipo de malware que busca la obtención de beneficio económico mediante estafa, utilizando técnicas de ingeniería social para causar un estado de shock o ansiedad en el usuario, causándole una alarma ante una amenaza, como el informarle y simularle la presencia de diversos tipos de malware en su máquina. El «rogueware» se ofrece entonces desinfectar la máquina, impresionando ser una solución antivirus. Su único objetivo una vez instalado en la máquina del usuario es requerir la activación del supuesto producto a través de diversas formas de pago. Algunas formas de spyware y de adware también usan las tácticas del scareware.

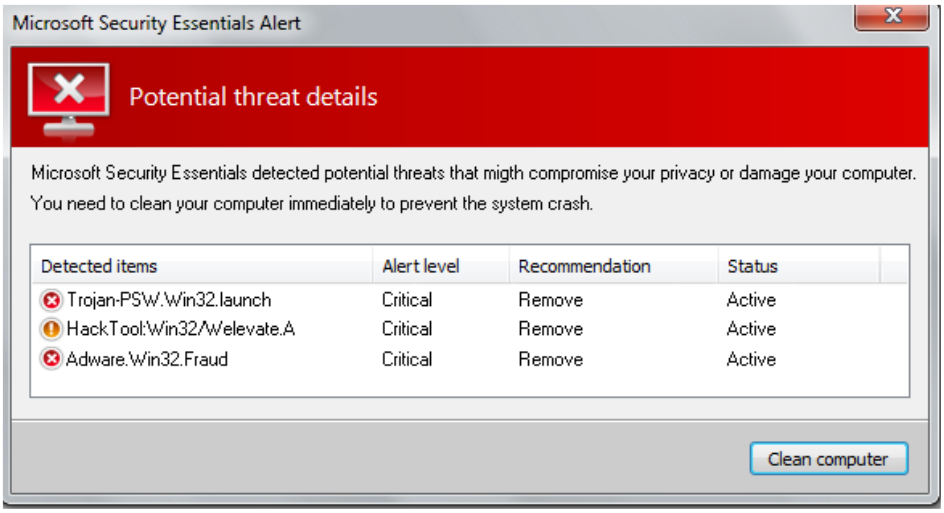

Figura 2 Malware tipo scaraware. Extraída de (INTECO, 2012)

2.2.1.9 Advanced Persistent Threat (APT)

Tipo del malware que consiste en un tipo sofisticado de ciberataque organizado, persistente y de larga duración. Constituye uno de los desafíos de seguridad más importantes y peligrosos, que deben afrontar hoy en día las organizaciones. Diseñado específicamente para acceder y obtener información de los sistemas de la organización objetivo, los vectores de ataque que usan pueden ser no muy diferentes de los empleados en otros tipos de ataque o incluso ser desarrollos específicos. Aprovechan vulnerabilidades conocidas o de día cero de los sistemas y aplicaciones de tecnologías de la información y comunicaciones, combinadas con técnicas de ingeniería social para explotar las debilidades o vulnerabilidades de la naturaleza humana.

2.2.2 Análisis de Malware

El análisis de malware proporciona la capacidad de analizar y comprender el funcionamiento del código malicioso (troyanos, virus, rootkits, etc.) para poder evaluar los daños causados y valorar las intenciones y capacidades del atacante. Conocer la estructura, funcionamiento e interacción del malware, aportará una valiosa información, no solo para el diseño y desarrollo de contramedidas eficaces, sino que también para ayudar a conocer el origen de un ataque y evaluar la capacidad de detección de los sistemas de la organización, al objeto de tomar las acciones de respuesta necesarias y adecuadas. El grado de complejidad de las técnicas y el nivel de conocimiento necesario para analizar malware es proporcional al nivel de sofisticación del mismo, estas técnicas conocidas como técnicas de análisis y reingeniería de malware, pretenden facilitar la adquisición de conocimiento sobre el mismo de una manera sistemática y metodológica (Higuera, 2013).

La protección en los sistemas de TIC del malware es actualmente uno de los problemas de seguridad más importantes para las organizaciones y los individuos, en este sentido se considera de vital importancia analizar su estructura, funcionamiento e interacción del mismo, pues aportará una valiosa información, no solo para el diseño y desarrollo de contramedidas eficaces, sino que también para ayudar a conocer el origen de un

ataque y evaluar la capacidad de detección de los sistemas de la organización, a objeto de tomar las acciones de respuesta necesarias y adecuadas.

En concreto entre los beneficios que se obtendría de su conocimiento tendríamos:

- Conocer el origen de un ataque e identificar al intruso.
- Evaluar la capacidad de detección de malware de los sistemas de protección de las organizaciones.
- Evaluar los daños causados por la intrusión y acciones del malware.
- Descubrir otras máquinas que han sido afectadas por el mismo malware.
- Identificar la vulnerabilidad que fue aprovechada por el malware, para obtener la actualización del software que la mitigue, si está disponible.
- Obtención de datos necesarios para poder implementar las defensas necesarias para mitigar y neutralizar los daños producidos por el malware particular analizado, entre las que se pueden incluir reglas de cortafuegos, de sistemas de detección de intrusiones tipo red y host y antivirus.
- Determinar el nivel de sofisticación y complejidad del malware.

Lo anterior exige la implementación de un entorno de pruebas o laboratorio, aislado de otras redes para poder asegurar la no propagación del malware o sus efectos a las redes de producción o incluso Internet. Este entorno debe disponer las capacidades de realizar línea base de la configuración de los equipos, clasificación y ejecución del malware, recogida y análisis de datos, análisis dinámico en un ambiente que simule su entorno real y el análisis estático de los ficheros malware.

Este análisis proporcionará un completo conocimiento del ciclo de vida del malware; su comportamiento, métodos de ocultación y ofuscación, sistema de actualizaciones y comunicaciones, características de los empaquetadores, características específicas de los ficheros entre otras.

2.2.2.1 Obtención del malware

Antes de comenzar con las actividades de análisis de malware, es necesario el disponer de mecanismos de obtención del mismo, entre los que tenemos los siguientes:

- Solicitarlo a instituciones de investigación que almacenan bases de datos de malware para sus estudios.
- Descargarlo de páginas de Internet especializadas.
- Capturárlo usando honeynets, honeypots.
- Capturándolo en una máquina infectada de la organización, al visitar servidores web maliciosos o debido a archivos adjuntos de correo electrónico, esto es más un incidente de seguridad a resolver por la organización que un medio de obtención, pero en varias ocasiones es el punto de partida para los análisis de malware.
- Utilizar un motor de búsqueda como Google, para buscar archivos binarios de malware.

Una forma de estudiar los métodos y patrones de ataque del malware, que permite conocer con detalle las vulnerabilidades de las redes de una organización y los ataques que las explotan para acceder a los sistemas protegidos, consiste en la implantación de honeypots, honeytokens y honeynets, como medida proactiva de defensa.

Este tipo de sistemas permite la obtención de malware para utilizarlo con propósito de investigación, pues permite obtener datos directamente de ataques reales al dejar de algún modo «expuestos» sistemas que puedan posteriormente analizarse. Estos sistemas deben ser construidos con el propósito de hacer creer al atacante que está ante un sistema real en producción, con aparentes problemas de seguridad debidos a una instalación incorrecta o una mala política de parcheo. Por supuesto esta apariencia no es real, por cuanto los sistemas estarán monitorizados y cualquier acceso será registrado en todos y cada uno de los movimientos que los atacantes realicen.

Los honeypots, honeytokens y honeynets deben ser o simular sistemas reales, con servicios y accesos reales o simulados a los mismos y con honeytokens falsos como documentos rastreables, que permitan un seguimiento de los mismos. De este modo, se podrá conseguir no solamente estudiar los métodos de ataque a los sistemas, sino que también los de distribución de información adquirida de forma ilegítima.

Este tipo de sistemas deben ser asegurados progresivamente de menos a más a fin de captar la atención del mayor número posible de atacantes, con el objetivo de poder captar cada vez patrones de ataque más avanzados. Los objetivos de este tipo de sistemas serían los siguientes:

- Implantación de medidas de seguridad proactiva que permita identificar nuevas tendencias en ataques y su evolución.
- Conseguir un conocimiento detallado de comportamiento de los intrusos así como sus motivos y las herramientas que utilizan.
- Ayudar a identificar las amenazas existentes y las técnicas de evasión utilizadas ante medidas de seguridad crecientes.
- Implementar contramedidas efectivas.

2.2.2.2 Análisis Estático

El análisis estático se realiza con dos enfoques, primero en la realización de un análisis de código ensamblador del malware, navegando a través de él al objeto de conseguir una mejor comprensión y funcionamiento del mismo, sin tener que ejecutar el archivo; el segundo enfoque se orienta hacia el análisis de las características del fichero que contiene el malware, que en este proyecto las denominamos **características estáticas**.

El análisis estático del código se basa en la utilización de técnicas de ingeniería inversa cargando el ejecutable en un desensamblador y mirando a las instrucciones del programa con el fin de descubrir lo que hace y ver cómo se comportaría en condiciones inusuales. Por ejemplo se pueden descubrir eventos que hagan que el malware se ejecute a partir de la ocurrencia de otros, como la visita a la página web del banco, cierta fecha, etc. Un punto importante es descubrir cómo pueden ser detectados por los programas antivirus o cómo pueden eludir cortafuegos y otras protecciones de

seguridad. El análisis estático ayuda a revelar lo que es capaz de hacer, cómo detenerlo y conocer sus características estáticas, es decir aquellas características que no dependen de la ejecución del fichero.

Aunque no hay una regla fija, la experiencia de los autores lleva a realizar el análisis estático en primer lugar, seguido por el análisis dinámico ya que la información recopilada durante el análisis estático suele proporcionar la información necesaria para el análisis dinámico; de acuerdo a los objetivos de los investigadores pueden realizar únicamente uno de los dos tipos de análisis.

Para realizar el análisis estático, se debe transformar el lenguaje ensamblador de nuevo al lenguaje de alto nivel que se utilizó para la creación del programa mediante técnicas llamadas ingeniería inversa con la ayuda de herramientas específicas.

En la se muestra el diagrama de este proceso:

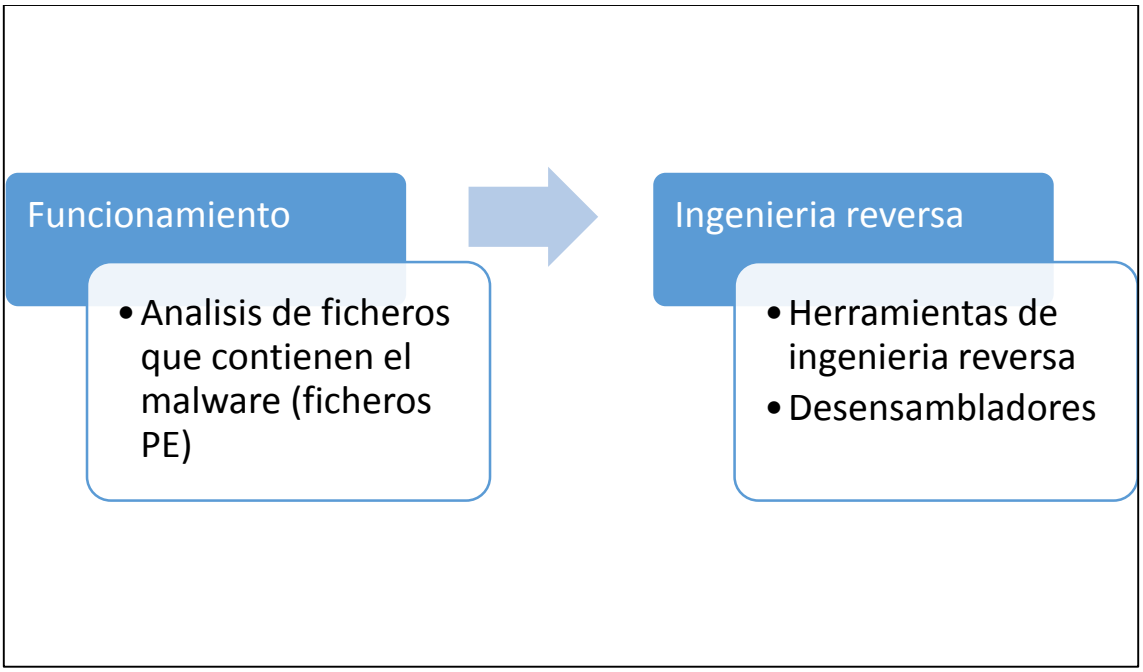

Figura 3 **Etapas del análisis estático de malware. Adaptado de (Higuera, 2013)**

2.2.2.3 Análisis Dinámico

Las técnicas básicas de análisis dinámico, implican la ejecución del malware en el sistema observando su comportamiento y los cambios que puedan ocurrir en el mismo. Se debe disponer de las herramientas necesarias para capturar las actividades y respuestas del malware, detectando en el sistema víctima las siguientes acciones:

- Detección de cambios del sistema de ficheros, por comparación.
- Acceso a los ficheros (obtenidos de los mensajes debug).
- Cambios en el registro (por comparación de ficheros).
- Accesos al registro (mensajes debug).
- Consultas DNS.
- Órdenes de mando y control.
- Byte de inicio de un socket (mediante un servidor TCP).

- Tráfico de red para comunicarse con otras máquinas (órdenes de mando y control).
- Descarga de archivos de Internet.

Debe tenerse en cuenta que cuando se realiza el análisis dinámico es crítico que el laboratorio de malware no esté conectado a otra red, es decir debe ser un entorno aislado.

En la Figura 4 se representa las tareas que comúnmente se realizan en este proceso, el paso inicial de esta fase es tomar una instantánea del sistema para que se pueda utilizar para comparar el estado del sistema antes y después de la ejecución del programa malicioso. Esto ayuda a identificar los archivos que se han agregado o modificado en el sistema y entender qué cambios han ocurrido después de la ejecución.

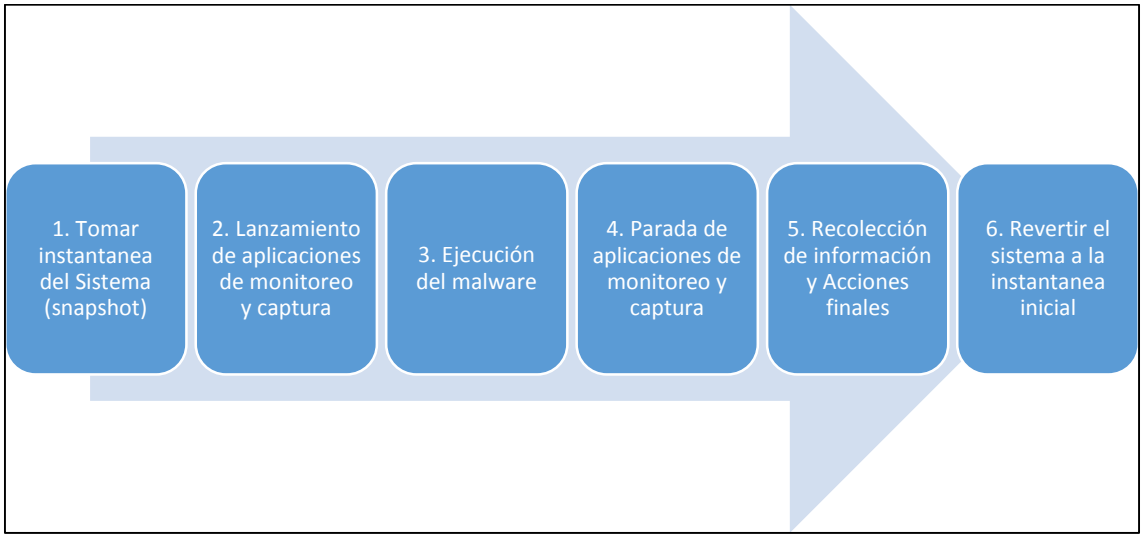

Figura 4 Proceso del análisis dinámico del malware. Adaptado de (Higuera, 2013)

Las tareas del análisis de malware dinámico pueden variar, como se mencionó anteriormente dependiendo del objetivo que tengan los investigadores y los factores que influyan en sus respectivas investigaciones.

2.3 Antecedentes

El presente trabajo presenta un análisis de las características estáticas para la clasificación del malware, por lo que es fundamental describir previamente un breve estado del arte sobre la clasificación del malware y las técnicas que actualmente se utilizan para esta clasificación.

2.3.1 Clasificación del malware

Con tantos tipos diferentes de malware (y la amplia variedad de programas de software malicioso que existen dentro de cada tipo), es importante que cada malware pueda clasificarse de forma precisa y diferenciarse fácilmente de otros programas maliciosos, así como también que las muestras de malware puedan clasificarse en clústers que representan las familias de los virus conocidos.

Actualmente ningún antivirus cuenta con un mecanismo de detección y clasificación totalmente efectivo y no es posible para las compañías antivirus abarcar las miles de muestras que reciben a diario. Se sabe que la mayoría de esas muestras son variaciones de malware que ya se tiene identificado y que pocos son completamente nuevos.

La clasificación del malware actualmente es el foco de muchas investigaciones. Con el desarrollo de la tecnología, el desarrollo del malware también ha debido innovarse y hoy en día existen diversas técnicas anti-análisis tales como:

- Cifrado; (carga útil cifrada)
- El polimorfismo; (carga útil encriptada, variando claves)
- El metamorfismo; (Instrucciones diferentes, misma funcionalidad)
- La ofuscación; (preservar la semántica)

La carga útil (más conocida como payload en inglés) es la parte del malware necesaria o el paquete de datos que se necesita para llevar a cabo el propósito del malware.

2.3.2 Técnicas de clasificación de malware

Para solucionar el problema de la clasificación del malware varios autores proponen distintos enfoques que generalmente están enfocados al análisis del comportamiento del malware, lo cual como se comentó en el apartado 2.2.2 se trataría de un análisis dinámico en entornos controlados, en este apartado se menciona brevemente algunos de estos enfoques:

"Learning and classification of malware behavior" (K. Rieck, 2008)

El enfoque de este artículo indica que la diversidad de malware y la cantidad de sus variantes debilitan severamente la efectividad de la detección basada en firmas clásica. Sin embargo, las variantes de las familias de malware comparten patrones de comportamiento típicos que reflejan su origen y propósito. Su objetivo es aprovechar estos patrones comunes para la clasificación del malware y proponer un método para el aprendizaje y la discriminación del comportamiento del malware. Su método se desarrolla en tres etapas:

a) el comportamiento del malware recolectado se controla en un entorno de sandbox,
b) con muestras de malware etiquetadas por un escáner anti-virus el clasificador comportamiento del malware es entrenado utilizando técnicas de aprendizaje y
c) las características discriminatorias de los modelos de comportamiento están clasificados para la explicación de decisiones de clasificación.

"Scalable, behavior based malware clustering" (U. Bayer, 2009)

Este enfoque en primer lugar ejecuta un análisis dinámico para obtener las trazas de ejecución de los programas de malware. Estas trazas de ejecución son entonces generalizadas en perfiles de comportamiento, que caracterizan la actividad de un programa en términos más abstractos. Los perfiles sirven como entrada a un algoritmo de agrupamiento eficiente que nos permite manejar conjuntos de muestras grandes, lo que denota la escalabilidad de este enfoque.

"FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors Operations and Abuse Reporting" (Rafique & Caballero, 2013)

Este enfoque presenta una herramienta a la que se le da un gran grupo de tráfico de la red obtenida mediante la ejecución de binarios de malware sin etiqueta y esta genera un clústers de los binarios de malware en familias y un conjunto de firmas de red para cada familia. FIRMA produce firmas de red para cada uno de los comportamientos de la red de una familia, independientemente del tipo de tráfico que utilice el malware (por ejemplo, HTTP, IRC, SMTP, TCP, UDP).

3Planteamiento del Problema

El desarrollo del proyecto surge como una necesidad de analizar si algunas de las características estáticas de los ficheros ejecutables pueden ayudar a mejorar las técnicas de clasificación de malware existentes, En este apartado primero se describe el Proyecto Malicia sobre el cual se realizará el análisis, y después se describen las características estáticas de un fichero ejecutable, que sirven de base para enfocarnos en una solución al problema, la cual se presenta en el apartado 4.

3.1 Malicia Project

Antonio Nappa, M. Zubair Rafique, y Juan Caballero, investigadores del Instituto Madrileño de Estudios avanzados IMDEA Software presentan el artículo "Driving in the Cloud: An Analysis of Drive-by Download Operations" (Nappa, Rafique, & Caballero, 2013) en el cual contribuyen con los siguientes aspectos:

- Proponen una técnica para identificar las operaciones no autorizadas mediante la agrupación de servidores exploit en función de su configuración y el malware que distribuyen.
- Informan sobre los aspectos de las operaciones de drive-by, como el número de servidores que utilizan, su infraestructura de alojamiento, su tiempo de vida, y las familias de malware que distribuyen.
- Se analiza el procedimiento de reporte de abuso, enviando informes sobre servidores exploit.
- Construyen una base de datos con el malware recogido, su clasificación y los metadatos asociados. Hacen esta base de datos disponible para otros investigadores.

La información sobre esta publicación y otras publicaciones relacionadas las presentan en "MALICIA PROJECT Malware in Cybercrime (Imdea Software Institute, 2013)". La base de datos de MALICIA consta de 11.688 binarios de malware recogidos de 500 servidores drive-by download, durante un período de 11 meses. Además de los propios binarios de malware, el conjunto de datos contiene una base de datos que detalla cuándo y desde dónde se recogió el malware, así como la clasificación del malware.

Para el desarrollo de este trabajo contamos con un total de 11.358 binarios de la base de datos de MALICIA, la información de los binarios que se encuentran en esta base de

datos ya se encuentra clasificada con sus respectivas familias en un 87,24%, teniendo aún 1.450 muestras de malware sin clasificar. En la Tabla 3 se presenta la clasificación actual, indicando cuantos binarios están en cada clúster de la clasificación de familias, en donde NULL indica los binarios que no tienen asignada una familia, se tiene un total de 53 familias.

FAMILIA	Número de malware por familia
winwebsec	5820
zbot	2168
NULL (binarios sin clasificar)	**1450**
zeroaccess	1306
securityshield	150
cridex	73
smarthdd	68
harebot	53
CLUSTER:85.93.17.123	45
cleaman	32
CLUSTER:newavr	29
CLUSTER:astaror	24
CLUSTER:46.105.131.121	20
CLUSTER:positivtkn.in.ua	14
CLUSTER:gdata	9
reveton	9
CLUSTER:chapterleomemorykombo.eu	9
spyeye-ep	7
CLUSTER:foreign	6
WinRescue	5
ramnit	5
CLUSTER:91.234.32.10	5
ransomNoaouy	5
CLUSTER:contacts	4
CLUSTER:mergenew	3

CLUSTER:autoit-plus	3
fakeav-webprotection	3
ufasoft-bitcoin	3
fakeav-rena	3
CLUSTER:online-police.com	2
cutwail	2
CLUSTER:fadedtext	1
CLUSTER:eikons.com	1
CLUSTER:mindamura.com	1
CLUSTER:azonpowzanadinoar.com	1
dprn	1
russkill	1
CLUSTER:justontime-12.com	1
CLUSTER:colorballs	1
CLUSTER:dzony3777.su	1
CLUSTER:in7cy	1
CLUSTER:bundlemonkey.com	1
CLUSTER:217.20.115.99	1
CLUSTER:mycomputer	1
CLUSTER:whatismyip.com	1
CLUSTER:blackandwhite	1
CLUSTER:paydayloatnsstc.ru	1
CLUSTER:184.82.148.49	1
CLUSTER:m9swachu.be	1
CLUSTER:price.dtdns.net	1
CLUSTER:floppies	1
CLUSTER:up2x.com	1
CLUSTER:halfmoon	1
CLUSTER:clarkclark	1

Tabla 3 Clasificación de familias de Malicia

El objetivo principal de este trabajo es mejorar el índice de clasificación analizando características estáticas que indiquen patrones o características comunes de los ficheros ejecutables y se puedan realizar clústers nuevos para clasificarlos en nuevas familias o añadir muestras de malware en las familias existentes.

3.2 Características estáticas de un PE

En el apartado 2.1 de este trabajo se presentó una descripción de los puntos destacables de un fichero Portable Ejecutable. En este apartado se mencionan las características que pueden ser de apoyo para proponer una solución al problema de la clasificación del malware.

Las características de los portables ejecutables se extraen de ciertas partes de los archivos PE (EXE o DLL). Estas son características significativas que podrían indicar que el archivo fue creado o infectado para realizar actividades maliciosas. Las características que se extraen se presentan en la Tabla 4.

Característica	Descripción	Ejemplo
Datos extraídos de la Cabecera del PE	describe la estructura física de un binario PE	el tiempo de creación / modificación, tipo de máquina, tamaño de archivo
Información opcional de las cabeceras del PE	describe la estructura lógica de un binario PE	la versión del vinculador, la sección de alineación, el tamaño del código, indicadores de depuración
Sección de importación	que DLL fueron importadas y que funciones de DLL se están utilizando	-
Sección de Exportaciones	que funciones se exportaron (si el archivo que se examina es una DLL)	-
Directorio de recursos	los recursos utilizados por un archivo determinado	cuadros de diálogo, cursores
Información de la versión	Metadatos del PE	el nombre interno y externo de un archivo, el número de versión

Tabla 4 Características estáticas de un PE

Las características estáticas de los ficheros PE que se presentan en la Tabla 4, podrían ser de ayuda para mejorar el índice de clasificación de MALICIA, pero varias de estas

características ya se encuentran en la base de datos y las que no se encuentran requerirían de un método de selección especializado (Asaf Shabtai, 2009) y estas características también pueden verse afectadas por las técnicas de desarrollo de malware mencionadas en el apartado 2.3.1.

Este antecedente plantea otro problema para seleccionar características estáticas que servirán para la clasificación del malware, en consecuencia el siguiente objetivo de este trabajo es el estudio de otras características estáticas más complejas o generadas a partir de algún algoritmo que mejoren el índice de clasificación de MALICIA.

4Solución adoptada

El presente trabajo cuenta con dos objetivos principales, primero seleccionar características estáticas que permitan realizar la clasificación del malware y a partir del cumplimiento de este objetivo el segundo que consistirá en mejorar el índice de clasificación de la base de datos de los binarios de MALICIA.

Para poder analizar en detalle las características de los ficheros ejecutables se ha utilizado varias herramientas como PEID para ver el detalle de las características internas de un fichero ejecutable, así como también las herramientas en línea VIRUSTOTAL y TOTALHASH. Estas dos últimas herramientas presentan varias características estáticas de las muestras de malware que analizan, incluyendo dos tipos de hashes que se puede obtener de un fichero ejecutable: Imphash y Pehash.

Estos hashes no son como los hashes comunes por ejemplo Sha1 o MD5 que realizan el algoritmo de la función hash sobre todo el fichero ejecutable, sino que toman porciones especificas del fichero ejecutable, lo cual solventaría de cierta forma el problema de la clasificación referente a las técnicas de desarrollo de malware presentadas en el apartado 2.3.1.

En esta sección se describen las dos características estáticas de ficheros ejecutables seleccionadas (Imphash y Pehash), las métricas que se utilizará para medir los índices de clasificación con estas características, y el proceso ejecutado para aplicar la clasificación sobre la base de datos de MALICIA.

4.1 Pehash

Georg Wicherski[2] presenta en su artículo "Pehash: A Novel Approach to Fast Malware Clustering" (Wicherski, 2013) un nuevo enfoque no criptográfico rápido para calcular la función hash para los archivos binarios en el formato Portable Ejecutable, el cual se describe en este apartado.

La función Pehash transforma la información estructural de una muestra en un valor hash. La agrupación de los binarios de valores hash calculados con la nueva función permite la detección de varias instancias de la misma muestra polimórfica, así como las

[2] https://www.usenix.org/legacy/events/leet09/tech/full_papers/wicherski/wicherski.pdf

muestras incompletas por ejemplo, debido a errores de transferencia en la comunicación no se completaron.

Las actuales soluciones anti-virus son demasiado lentas e imprecisas para escanear cada muestra de entrada a la lista negra y sobre la base de estos datos. Para lograr la agrupación rápidamente Georg Wicherski desarrollo una función hash que genera el mismo valor hash para cualquier dos instancias de la misma muestra, por lo tanto la función Pehash puede ser utilizada para el Clustering del malware.

4.1.1 Diseño de la función Pehash

Dado que el malware es a menudo empaquetado, específicamente el malware polimórfico utilizando stubs, sin suponer nada sobre el código real del malware, pero el stub puede ser hecho de forma estática además, el malware polimórfico o los empaquetadores a menudo insertan instrucciones aleatorias en los stubs, lo que de nuevo da como resultado un valor hash único para cada instancia de un espécimen. Las secciones de datos podría estar cifradas y con el código que se ejecuta se descifrarían en tiempo de ejecución, mirando a los contenidos de las secciones en un fichero PE no es factible conseguir la información necesaria para desarrollar un hash que permita agrupar al malware en clústers.

Como esta función hash debe ser de baja complejidad computacional para el binario malware promedio, sólo se puede usar un conjunto limitado de información de forma rápida obtenida o calculada. En concreto, emular el código del malware no es una opción aunque sólo sea parcialmente, ya que incluso pueden ocurrir en pequeños bucles stubs que se encuentren cifrados, de los cuales la longitud de tiempo de ejecución no puede ser fácil de predecir. Incluso si la emulación es limitada por un tiempo o límite de recuento de instrucciones, sería fácil anteponer un tiempo de inactividad relativamente largo a cualquier acción significativa, por lo que los resultados de la emulación no son útiles para el clustering.

4.1.2 Caracteristicas

Además de los datos con la información específica del fichero ejecutable, se debe tener en cuenta datos adicionales para tener suficiente distinción entre binarios. Los ejemplares de malware polimórfico de hoy comparten las mismas características estáticas de un PE ya que el código que genera nuevas instancias no realiza ninguna re-vinculación, sólo genera un nuevo stub para descifrar y hacer el re-cifrado de la nueva llave. Por lo tanto, las siguientes características de los ficheros portables ejecutables serán comunes en varios ejemplares del fichero:

Características: Indicadores generales para el Portable Ejecutable, por ejemplo, si el archivo dado es un archivo DLL o sólo se puede ejecutar en una sola máquina del procesador.

Subsistema: Indica que este fichero correrá los subsistema de Windows, como la GUI, CLI o el controlador de dispositivos.

Stack Commit Size: El tamaño inicial de la pila del programa que se asignará en bytes. Este valor se redondea a un valor divisible por 4096, límite de la página de Windows.

Heap Commit Size: Tamaño inicial de almacenamiento dinámico que se asignará al ejecutable, también redondeado al límite de tamaño de página.

Para cada sección en el Portable Ejecutable, la siguiente información estructural se incluye:

Dirección Virtual: La dirección, el contenido de la sección va a ser cargado en memoria o se asignará a una sección .bss.

Tamaño de la sección sin ejecutar: El tamaño de la sección en el propio archivo de ejecutables portables; puede ser más pequeño que el tamaño real que ocupa en la memoria después de la carga debido al redondeo a límites de la página

Características de la sección del PE .section: Indicadores de sección que describe los privilegios iniciales de la memoria asignada, como la lectura, la escritura y la ejecución de código. También contiene información acerca de la alineación y si la sección contiene datos no inicializado como una sección .bss.

Dado que no todos los valores de todos los bits de todos son realmente usados o contienen información útil, sólo partes seleccionadas de los valores se incluyen en el hash. Las modificaciones de estos bits significativos indican cambios significativos en el binario de malware, reduciendo el riesgo de que la utilidad de esta función hash para una misma métrica de hash sea destruida fácilmente por bits individuales cambiados. Los 8 bits superiores para la pila y cola de 32 bits tienen tamaños que casi siempre son cero. Por otra parte, algunos valores pueden diferir en los bits más bajos debido a los cambios de tamaño pequeño en el malware polimórfico y deben ser descartados para permitir la coincidencia exacta de los valores hash. El siguiente pseudocódigo describe la generación exacta para el valor hash a partir de las propiedades globales, donde v [8...24] significa bits 8 a 24 del valor de v y $\oplus$ significa XOR:

$$
\begin{aligned}
\mathit{hash}[0] &:= \mathit{characteristics}[0..7] \\
&\quad \oplus \ \mathit{characteristics}[8..15] \\
\mathit{hash}[1] &:= \mathit{subsystem}[0..7] \\
&\quad \oplus \ \mathit{subsystem}[8..15] \\
\mathit{hash}[2] &:= \mathit{stackcommit}[8..15] \\
&\quad \oplus \ \mathit{stackcommit}[16..23] \\
&\quad \oplus \ \mathit{stackcommit}[24..31] \\
\mathit{hash}[3] &:= \mathit{heapcommit}[8..15] \\
&\quad \oplus \ \mathit{heapcommit}[16..23] \\
&\quad \oplus \ \mathit{heapcommit}[24..31]
\end{aligned}
$$

Además, para cada sección, los siguientes sub-hash se anexan al hash:

$$\mathit{shash}[0] := \mathit{virtaddress}[9..31]$$

$$
\begin{aligned}
shash[2] &:= rawsize[8..31] \\
shash[4] &:= characteristics[16..23] \\
&\quad \oplus \ characteristics[24..31] \\
shash[5] &:= kolmogorov \in [0..7] \subset \mathbb{N}
\end{aligned}
$$

4.2 Imphash

El 23 de Enero del 2014 la compañía MANDIANT[3] especialidad en ofrecer productos y servicios de seguridad informática, publica en su blog el artículo "Tracking Malware with Import Hashing" (Mandiant Corporation, 2014) donde muestran uno de los métodos que ellos utilizan para identificar y clasificar el malware.

Una forma única que Mandiant da seguimiento a los grupos específicos de amenazas backdoors, es realizando un seguimiento de las importaciones de un fichero ejecutable PE. Las importaciones son las funciones que una pieza de software (en este caso, backdoor) utiliza para llamar a otros archivos (por lo general varias DLL que ofrecen funcionalidad para el sistema operativo Windows). Para realizar un seguimiento de estas importaciones, Mandiant crea un hash basado en nombres de biblioteca / API y su orden específico dentro del ejecutable. Nos referimos a este convenio como un "Imphash" (de "importación de hash"). Debido a la forma en que se genera la tabla de importación de un PE (y, por tanto, cómo se calcula su Imphash), podemos usar el valor Imphash para identificar muestras de malware relacionados. También podemos utilizarlo para buscar nuevas muestras similares que el mismo grupo de amenaza puede haber creado y utilizado.

La implementación de este hash ya se encuentra en la última versión del paquete complementario de Python Pefile por lo que en este trabajo no se implementará esta función, únicamente se la importara desde el paquete de Python mencionado.

4.3 Métricas

Para medir la calidad de los clusters que se realizarán en este trabajo se ha elegido las siguientes métricas que se describen a continuación:

4.3.1 RS, JC y FM

El clustering se puede ver como un proceso de aprendizaje sin supervisión en un conjunto de datos para los que el clustering de referencia por lo general no está disponible. Por lo tanto, a diferencia de los entornos de aprendizaje supervisado, el análisis de la validez de los resultados del Clustering es intrínsecamente difícil. La evaluación de la calidad de los resultados de la agrupación a menudo implica el uso de criterios subjetivos de optimalidad, que suelen ser aplicaciones específicas, y por lo general implica un extenso análisis manual por expertos en el dominio. Para ayudar en el proceso de validación de clústeres, varios métodos e índices de calidad se han

[3] https://www.mandiant.com/blog/tracking-malware-import-hashing/

propuesto (M. Halkidi, 2001), cuyo objetivo es evaluar los resultados de la agrupación para encontrar la partición que mejor se adapte a los datos (Perdisci & ManChon, 2008).

Suponiendo que un clustering de referencia está disponible, se describen los siguientes índices de validez externos que pueden ser utilizados para medir la calidad de los resultados de la agrupación. Sea M el conjunto de datos R_c = {R_{c1},.., R_{cs}} el conjunto de las agrupaciones de referencia s, y C = {C_1,.., C_n} ser nuestros resultados de la agrupación terminado. Dado un par de muestras de datos (M_1, M_2), con m_1, m_2 ∈ M, podemos calcular las siguientes cantidades:

- **a** es el número de pares (M_1, M_2) para los que si ambas muestras pertenecen al mismo grupo de referencia R_{ci}, también pertenecen al mismo grupo C_j.
- **b** es el número de pares (M_1, M_2) para el que ambas muestras pertenecen al mismo grupo de referencia Rci, pero se asignan a dos grupos diferentes de C_k y C_h.
- **c** es el número de pares (M_1, M_2) para el que ambas muestras pertenecen al mismo grupo Ci, pero se asignan a dos grupos de referencia diferentes R_{ck} y R_{ch}.
- **d** es el número de pares (M_1, M_2) para que si las muestras pertenecen a dos grupos de referencia diferentes R_{Ci} y R_{Cj}, ellos también pertenecen a diferentes grupos C_l y C_m.

Basándonos en las definiciones anteriores se puede calcular las siguientes métricas de validación de clustering

- **Rand Statistic (RS).** $RS = \frac{a+d}{a+b+c+d} = \frac{a+d}{|M|}$
- **Jaccard Coefficient (JC).** $JC = \frac{a}{a+b+c}$
- **Folkes and Mallows Index (FM).** $FM = \frac{a}{\sqrt{(a+b)(a+c)}}$

Para los tres índices RS, JC, FM, que tienen valores entre [0, 1], cuanto mayor sea el valor, mayor es la similitud entre el clustering C y la agrupación de referencia R_c.

4.3.2 Precision

Al implementar algoritmos de clustering Intuitivamente, nos esforzamos por alcanzar resultados en los que cada grupo contiene sólo los elementos de un tipo particular (U. Bayer, 2009). El objetivo de la precisión (Precision) es medir la eficacia de un algoritmo de clustering distinguiendo entre las muestras que son diferentes. Es decir, la precisión captura como de bien un algoritmo de clustering asigna muestras de diferentes tipos para diferentes grupos. Más formalmente, la precisión se define de la siguiente manera: Supongamos que tenemos un clustering de referencia T = T1, T2, . . , Tt con t clústers y un clustering C = C1, C2, . . , Cc con c clústers (para un conjunto de muestra A = a1, a2, . . , an). Por cada Cj ∈ C, calculamos el valor Precision del clúster como:

$$P_j = max\left(\left|C_j \cap T_1\right|, \left|C_j \cap T_2\right|, \dots, \left|C_j \cap T_t\right|\right)$$

El valor total de Precision es:

$$\mathrm{P} = \frac{(|C_1|P_1 + |C_2|P_2 + \cdots + |C_c|P_c)}{\mathrm{n}}$$

Donde n es el número total de elementos.

4.3.3 Recall

Claramente, preferimos una agrupación donde todos los elementos de un tipo se asignen al mismo grupo (U. Bayer, 2009). El Recall se utiliza para medir qué tan bien un algoritmo de agrupamiento reconoce muestras similares. Es decir, el Recall capta lo bien que un algoritmo asigna muestras del mismo tipo al mismo grupo. Definimos formalmente el Recall de la siguiente manera: Supongamos que tenemos una agrupación de referencia T = T1, T2, . . , Tt con t clústers y un clustering C = C1, C2, . . , Cc con c clústers. Para cada Tj ∈ T, calculamos el valor Recall del clúster como:

$$R_j = max\left(\left|C_1 \cap T_j\right|, \left|C_2 \cap T_j\right|, \ldots, \left|C_c \cap T_j\right|\right)$$

El valor total de Precision es:

$$\mathrm{R} = \frac{(|T_1|R_1 + |T_2|R_2 + \cdots + |T_t|R_t)}{\mathrm{n}}$$

Donde n es el número total de elementos.

El algoritmo de clasificación primitivo que crea un clúster para cada muestra alcanza una precisión óptima, pero el peor Recall. El algoritmo que combina todas las muestras en un solo grupo, en cambio, logra un Recall óptimo, pero el peor Precision. En la práctica, un algoritmo debe proveer tanto de alta Precision y Recall. Es decir, cada grupo debe contener todas las muestras de un tipo, pero no más.

4.3.4 F-Measure

El algoritmo de clasificación primitivo que crea un clúster para cada muestra alcanza una precisión óptima, pero el peor Recall. El algoritmo que combina todas las muestras en un solo grupo, en cambio, logra un Recall óptimo, pero el peor Precision. En la práctica, un algoritmo debe proveer tanto de alta Precision y Recall. Es decir, cada grupo debe contener todas las muestras de un tipo, pero no más.

El F-Measure es una métrica que combina el Precision y el Recall es la media armónica de Precision y el Recall.

El valor de F-Measure es:

$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

4.4 Clasificación de MALICIA

Una vez que se ha analizado las características estáticas de los ficheros ejecutables que se utilizará para cumplir con los objetivos de este trabajo, el siguiente paso fue preparar un entorno seguro en el que se pueda realizar todas las tareas necesarias para clasificar todas las muestras de malware.

4.4.1 Entorno de trabajo

Para realizar todas las actividades prácticas del análisis se implementó un entorno virtual con las respectivas medidas de seguridad, esto indica que se tendría un equipo host y uno virtual. Las características principales del equipo host son las siguientes:

Característica	Especificación
Modelo del equipo	Dell Inspiron 14R
Procesador	Intel Core i7 a 2,5 GHz
Memoria RAM	16 GB
Sistema Operativo	Windows 8.1 Pro
Almacenamiento	1TB

Tabla 5 Características equipo host

Sobre el equipo host se implementó una máquina virtual con las siguientes características principales:

Característica	Especificación
Modelo del equipo	VM Ware Player
Procesador	Intel Core i7 a 2,5 GHz
Procesadores Lógicos	4
Memoria RAM	8 GB
Sistema Operativo	Ubuntu 14.4 LTS
Almacenamiento	100 GB

Tabla 6 Características Entorno de trabajo

Una vez que se tenía el equipo virtual con el sistema operativo listo y actualizado se procedió a preparar el entorno, para contar con las herramientas necesarias. Para esto se instaló los siguientes paquetes:

- MySql server
- MySql Workbench
- Python 2.7
- Librerías de Python
    - Pefile
    - hashlib
    - bitstring
    - bz2
    - MySQLdb
- Vim

Una vez que el entorno de trabajo estaba listo, se levantó la base de datos MALICIA la que se puede obtener con fines investigativos en el sitio web del proyecto[4].

4.4.2 Generación de Imphash y Pehash de los PE

Se realizó un script en Python para obtener el Pehash y el Imphash de los ficheros ejecutables. Estos ficheros actualmente tienen como nombre del archivo el hash SHA1, En la base de datos MALICIA en la tabla FILES se tiene a información de los ficheros binarios, los atributos que podemos encontrar en esta tabla son los siguientes:

- FILE_ID
- SHA1
- SIZE
- FILETYPE
- PACKER
- IN_STORE
- ICON
- ICON_SIZE
- ICON_LABEL
- FAMILY
- TRAFFIC_LABEL
- TRAFFIC
- AFFILIATE
- SSHOT_LABEL

Con el Script de la Tabla 7 se generó en un archivo de texto con la información almacenada con la estructura <<Nombre del archivo>>, <<MD5>>, <<SHA1>>, <<Pehash>>, <<Imphash>>.

```
#!/usr/bin/python

"" Author: Richard Rivera

    Script for obtain the md5, sha1, Imphash, Pehash for a specific
directory

    Execution example: $ python Hashes.py

    Hashes.py don't need parameters


from __future__ import division

import os

import sys

import hashlib

import pefile
```

[4] http://malicia-project.com/

```
import bitstring
import string
import bz2


def recursive():
    with open("output", "w") as a:
        for path, subdirs, files in os.walk ('/home/binarios/'):
            for filename in files:
                #md5
                with open(os.path.join(path,filename),'r') as f:
                    for chunk in iter(lambda: f.read(), b""):
                        md5 = hashlib.md5()
                        md5.update(chunk)
                md5Hash=md5.hexdigest()
                #sha1
                with open(os.path.join(path,filename),'r') as f:
                    for chunk in iter(lambda: f.read(), b""):
                        sha1= hashlib.sha1()
                        sha1.update(chunk)
                sha1Hash=sha1.hexdigest()
                #Imphash
                try:
                    p = pefile.PE(os.path.join(path,filename))
                    Imphash = p.get_imphash()
                    #control for PE file with empty import table
                    if Imphash=="":
                        Imphash="null"
                except:
                    #control for not PE file
                    pass
                    Imphash="null"
                #Pehash to file
                try:
Pehash=pehash_To_File(os.path.join(path,filename))
```

```
                except:
                    pass
                    Pehash="null"
                #print to file
<<filename>>,<<md5>>,<<sha1>>,<<Imphash>>,<<Pehash>>
                f =
("{},{},{},{},{}".format(os.path.join(filename),md5Hash,sha1Hash,imp
hash,pehash))
                    a.write(str(f) + os.linesep)
def main(args):
    recursive()
if __name__=="__main__":
   main(sys.argv)
def pehash_To_File(filename):
""
Script published by Totalhash.com as pehash.py
adapted as a function for Hashes.py by Richard Rivera
""
```

Tabla 7 Script para generar el Imphash y Pehash de un PE

El script anterior utiliza el script de Pehash publicado por TotalHash[5], el que fue adaptado para ser utilizado como una función dentro del script anterior, este Script completo se presenta en el Anexo 1. Una vez que se tenía el archivo de salida se añadió estos nuevos atributos en la base de datos de MALICIA.

Al generar los Pehash e Imphash de los binarios de MALICIA, se encontró algunas novedades presentadas en la Tabla 8 que debieron ser analizadas:

Ficheros PE analizados	Ficheros PE defectuosos	Ficheros PE validos	Ficheros PE sin Imphash	Clústers de Imphash	Clústers de Pehash
11363	5	11358	133	2028	1500

Tabla 8 Resultados del Script de Imphash y Pehash

En un principio se contaba con 11363 binarios, debido a que 5 ficheros PE se marcaban como que no eran ficheros PE, se los analizo individualmente y se optó por retirar estos binarios y reducir la muestra a 11358 ficheros PE, de esta muestra se notó que 133 binarios tenían la tabla de importación vacía, y se generaron 2028 distintas cadenas de Pehash que se los llamo los clústers de Imphash y de la misma forma se obtuvieron 1500 clústers de Pehash.

[5] http://totalhash.com/

4.4.3 Métricas para Imphash y Pehash

Una vez que se conocía el número de clústers que tienen estas 2 características estáticas de los ficheros PE de MALICIA, se procedió a obtener las métricas comentadas en el apartado 4.3 de este trabajo, para conocer cuál sería la mejor forma de aplicar la clasificación sobre los binarios de MALICIA para esto se implementaron dos Scripts en Python que nos permitían obtener las variables a, b, c, d necesarias para los primeros parámetros estudiados.

4.4.3.1 Métricas RS, JC, FM

El primer script "pehashparameters.py" presentado en la Tabla 9 obtiene las variables a, b, c, d para calcular el RS, JC, FM; como se mencionó anteriormente en la base de datos local que cargo los datos de MALICIA ya se añadieron los valores de Imphash y el Pehash para los binarios de malicia. Este script crea una matriz para realizar las comparaciones necesarias de las métricas que se quiere obtener, los valores de esta matriz los toma directamente desde la base de datos, tomando en cuenta los datos de la Tabla 8 solo se toman como ficheros PE validos aquellos que cuentan con el valor de Pehash.

Para obtener los índices de validez de los clústers se utilizan las métricas RS, JC, FM, estas métricas necesitan de dos clústers, el clúster de referencia Rc. y el clúster de resultados C., en nuestro caso el Rc., serán los clústers que se tienen previamente de la clasificación de acuerdo a familias de malware que tiene MALICIA, y el C., serán los clústers que se generaron de la obtención del Imphash y el Pehash, teniendo en cuenta esto se analizó que para obtener mejores resultados se debe aplicar estas métricas para el conjunto de ficheros PE de malicia que si cuentan con el atributo de familia. Con lo cual la muestra de ficheros PE se reduce a 9908.

```
#!/usr/bin/python

# -*- coding: utf-8 -*-

"" Author: Richard Rivera

    Script for parameters for malware Clustering

""

import MySQLdb as mdb

import sys


db = mdb.connect('localhost', 'user', 'passsword', 'RELEASE1.0')


cursori = db.cursor()

cursorj= db.cursor()

# execute SQL select statement

cursori.execute("select FAMILY, PEHASH, FILE_ID from FILES where
IN_STORE='1' and PEHASH IS NOT NULL AND FAMILY IS NOT NULL")
```

```
a=0.0

b=0.0

c=0.0

d=0.0

totalMatrix=0.0

for rowi in cursori.fetchall():

    cursorj.execute("select FAMILY, PEHASH, FILE_ID from FILES where
IN_STORE= '1' and PEHASH IS NOT NULL AND FAMILY IS NOT NULL")

    for rowj in cursorj.fetchall():

        if rowi[2]<rowj[2]:

            totalMatrix=totalMatrix+1

            # this if calculate the a parameter

            if (rowi[0]==rowj[0] and rowi[1]==rowj[1]and
rowi[0]!=None):

                a=a+1

            # this if calculate the b parameter

            elif(rowi[0]==rowj[0]and rowi[1]!=rowj[1]and
rowi[0]!=None):

                b=b+1

            # this if calculate the c parameter

            elif((rowi[0]!=rowj[0]and rowi[1]==rowj[1])or
(rowi[0]==None and rowi[1]==rowj[1])):

                c=c+1

            # this if calculate the c parameter

            elif((rowi[0]!=rowj[0]and rowi[1]!=rowj[1])or
(rowi[0]==None and rowi[1]!=rowj[1])):

                d=d+1


parameters =("TotalMatrix= {}, A= {} B={} C={}
D={}".format(totalMatrix,a,b,c,d))

print parameters

db.close()
```

Tabla 9 Script pehashparameters.py

De manera similar se crea el Script "imphashhashparameters.py" que se presenta en la

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"" Author: Richard Rivera
    Script for parameters for malware Clustering
""
import MySQLdb as mdb
import sys


db = mdb.connect('localhost', 'user', 'password', 'RELEASE1.0')
cursori = db.cursor()
cursorj= db.cursor()
# execute SQL select statement
cursori.execute("select FAMILY, IMPHASH, FILE_ID from FILES where
IN_STORE='1' and PEHASH IS NOT NULL AND FAMILY IS NOT NULL")
a=0.0
b=0.0
c=0.0
d=0.0
totalMatrix=0.0
for rowi in cursori.fetchall():
    cursorj.execute("select FAMILY, IMPHASH, FILE_ID from FILES
where  IN_STORE='1' and PEHASH IS NOT NULL AND FAMILY IS NOT NULL")
    for rowj in cursorj.fetchall():
        if rowi[2]<rowj[2]:
            totalMatrix=totalMatrix+1
            # this if calculate the a parameter
            if (rowi[0]==rowj[0] and rowi[1]==rowj[1]and
rowi[0]!=None):
                a=a+1
            # this if calculate the b parameter
            elif(rowi[0]==rowj[0]and rowi[1]!=rowj[1]and
rowi[0]!=None):
                b=b+1
            # this if calculate the c parameter
            elif((rowi[0]!=rowj[0]and rowi[1]==rowj[1])or
(rowi[0]==None and rowi[1]==rowj[1])):
```

```
                c=c+1

            # this if calculate the c parameter

            elif((rowi[0]!=rowj[0]and rowi[1]!=rowj[1])or
(rowi[0]==None and rowi[1]!=rowj[1])):

                d=d+1


parameters =("TotalMatrix= {}, A= {} B={} C={}
D={}".format(totalMatrix,a,b,c,d))

print parameters

db.close()
```

Tabla 10 Script imphashparameters.py

Con estos dos Scripts ya se pueden obtener las métricas RS, JC, FM que se presentan en la Tabla 11.

Característica	a	b	c	d	RS	JC	FM
Imphash	181987	19972611	1298	28923382	59,30%	0,90%	9,47%
Pehash	394723	19759875	4	28924676	59,74%	1,96%	13,99%

Tabla 11 Resultado de Parámetros RS, JC, FM

Para comprender con más claridad los valores de las variables a, b, c, d que se presentan en la tabla anterior se debe tomar en cuenta que contamos con 9908 ficheros binarios, y como se mencionó la idea conceptual del script es realizar una matriz de comparación por lo tanto la suma de las variables a, b, c, d debe ser igual al resultado de la siguiente formula:

$$T = N(N-1)$$

Donde N será el número total de ficheros binarios utilizados en esta matriz de comparación, una matriz de comparación generaría N*N resultados, pero no se debe tomar en cuenta el par de comparación de un elemento consigo mismo:

$$T = N(N-1)$$
$$T = 9908(9908-1)$$
$$T = 49079278$$
$$T = a+b+c+d$$
$$T = 181987 + 19972611 + 1298 + 28923382$$
$$T = 49079278$$

Se puede notar que en los dos casos T tiene el mismo valor por lo que los valores serían correctos, se obtiene el mismo resultado con los valores de las variables a, b, c, d para Pehash.

Además de los resultados de las métricas RS, JC, FM, algo importante que se pudo obtener con estos scripts es el resultado de la variable c la cual nos indica que de la clasificación actual de las familias de MALICIA hay ficheros PE que están asignados en una familia con la que no comparten estas características estáticas. Esto será parte de un futuro análisis que llevarán a cabo en el proyecto MALICIA, pero como parte de este trabajo se decidió obtener el detalle de cuáles son estos ficheros binarios que estarían generando el valor de la variable c. Para esto se modificó levemente el script de la Tabla 9 y la Tabla 10 para imprimir a un fichero los datos de los binarios que provocan este valor. Para el atributo Pehash se presentan en la Tabla 12. Para los correspondientes al atributo Imphash se presentan en el Anexo 2.

FICHERO 1		FICHERO 2		ATRIBUTO COMPARTIDO
FILE_ID	FAMILY	FILE_ID	FAMILY	PEHASH
13381	cridex	13427	zeroaccess	e4852bafdd7d77aa5e8116106bb02d3e9a850166
13397	cridex	13427	zeroaccess	e4852bafdd7d77aa5e8116106bb02d3e9a850166
13415	cridex	13427	zeroaccess	e4852bafdd7d77aa5e8116106bb02d3e9a850166
13422	cridex	13427	zeroaccess	e4852bafdd7d77aa5e8116106bb02d3e9a850166

Tabla 12 Ficheros PE de Malicia con igual Pehash y distinta familia

Esta tabla nos indica que al comparar el fichero 1 con el fichero 2, estos comparten el atributo Pehash pero actualmente están asignados a distintas familias, es decir se puede notar que 4 elementos de la familia cridex comparten el Pehash con un elemento de la familia zeroaccess.

Como se mencionó en el apartado 4.3 estas métricas nos indicará el índice de validez de la clasificación, y posibles errores en el "Clustering de Referencia", pero para tener indicadores que permitan una mejor interpretación se obtendrá también las métricas Precision y Recall.

4.4.3.2 Métricas Precision y Recall

De igual forma que para las métricas anteriores para obtener el Precision y el Recall se realizaron 2 Scripts por cada uno de estas métricas, generando los siguientes Scripts

- PrecisionImphash.py Tabla 13
- PrecisionPehash.py Tabla 14
- RecallImphash.py Tabla 15
- RecallPehash.py Tabla 16

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"" Author: Richard Rivera
    Script for parameters for malware Clustering
""
import MySQLdb as mdb
import sys
```

```
db = mdb.connect('localhost', 'user', 'password', 'RELEASE1.0')
cursorC = db.cursor()
cursorT= db.cursor()
cursorP=db.cursor()
cursorN=db.cursor()
#getting n from the valid samples IS THE SAME FROM
PRECISIONPEHASH.PY BECAUSE THE VALID SAMPLES ARE SAMPLES WITH PEHASH
cursorN.execute("SELECT COUNT(FILE_ID) FROM FILES where PEHASH IS
NOT NULL and FAMILY IS NOT NULL")
(N,)=cursorN.fetchone()
print N
Precision=0.0
P=0.0
Pj=0.0
cursorC.execute("SELECT DISTINCT IMPHASH FROM FILE_TMPC")
for rowC in cursorC.fetchall():
    cursorT.execute("SELECT DISTINCT FAMILY FROM FILES WHERE FAMILY
IS NOT NULL AND PEHASH IS NOT NULL AND IMPHASH IS NOT NULL")
    maxJ=0
    for rowT in cursorT.fetchall():
        cursorP.execute("SELECT COUNT(FAMILY) FROM FILES WHERE
IMPHASH='%s'AND FAMILY='%s'"%(rowC[0],rowT[0] ))
        (cP,)=cursorP.fetchone()
        if maxJ<cP:
            maxJ=cP
    Pj=Pj+maxJ
#ALL Pj with Imphash null will generate an Pj=1 then
#--
cursorImphashNull=db.cursor()
cursorImphashNull.execute("SELECT COUNT(FILE_ID) FROM FILES WHERE
FAMILY IS NOT NULL AND PEHASH IS NOT NULL AND IMPHASH IS NULL")
(cursorIN,)=cursorImphashNull.fetchone()
Pj= Pj+cursorIN
#--
print Pj
Precision=float(Pj)/N*100
print "%2f"%(Precision)
db.close()
```

Tabla 13 Script para obtener el Precision de Imphash

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"" Author: Richard Rivera
    Script for parameters for malware Clustering
""
import MySQLdb as mdb
import sys

db = mdb.connect('localhost', 'user', 'password', 'RELEASE1.0')

cursorC = db.cursor()
cursorT= db.cursor()
cursorP=db.cursor()
cursorN=db.cursor()
#getting n from the valid samples. records with Pehash are the valid
samples
cursorN.execute("SELECT COUNT(FILE_ID) FROM FILES where PEHASH IS
NOT NULL and FAMILY IS NOT NULL")
(N,)=cursorN.fetchone()
print N
```

```
Precision=0.0
P=0.0
Pj=0.0
cursorC.execute("SELECT DISTINCT PEHASH FROM FILES where PEHASH IS
NOT NULL and FAMILY IS NOT NULL")
for rowC in cursorC.fetchall():
    cursorT.execute("SELECT DISTINCT FAMILY FROM FILES where PEHASH
IS NOT NULL and FAMILY IS NOT NULL")
    maxJ=0
    for rowT in cursorT.fetchall():
        cursorP.execute("SELECT COUNT(FAMILY) FROM FILES WHERE
PEHASH='%s'AND FAMILY='%s'"%(rowC[0],rowT[0] ))
        (cP,)=cursorP.fetchone()
        if maxJ<cP:
            maxJ=cP
    Pj=Pj+maxJ
print cursorC.rowcount
print Pj
Precision=float(Pj)/N*100
print "%2f"%(Precision)
db.close()
```

Tabla 14 Script para obtener el Precision de Pehash

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"" Author: Richard Rivera
    Script for parameters for malware Clustering
""
import MySQLdb as mdb
import sys

db = mdb.connect('localhost', 'user', 'password', 'RELEASE1.0')

cursorC = db.cursor()
cursorT= db.cursor()
cursorR=db.cursor()
cursorN=db.cursor()
#getting n from the valid samples. records with Pehash are the valid
samples
cursorN.execute("SELECT COUNT(FILE_ID) FROM FILES where PEHASH IS
NOT NULL and FAMILY IS NOT NULL")
(N,)=cursorN.fetchone()
print N
Recall=0.0
R=0.0
Rj=0.0
cursorC.execute("SELECT DISTINCT FAMILY FROM FILES where PEHASH IS
NOT NULL and FAMILY IS NOT NULL")
for rowC in cursorC.fetchall():
    cursorT.execute("SELECT DISTINCT IMPHASH FROM FILES where PEHASH
IS NOT NULL and FAMILY IS NOT NULL AND IMPHASH IS NOT NULL")
    maxJ=0
    for rowT in cursorT.fetchall():
        cursorR.execute("SELECT COUNT(IMPHASH) FROM FILES WHERE
IMPHASH='%s'AND FAMILY='%s'"%(rowT[0],rowC[0] ))
        (cR,)=cursorR.fetchone()
        if maxJ<cR:
            maxJ=cR
    Rj=Rj+maxJ
print cursorC.rowcount
```

```
print Rj
Recall=float(Rj)/N*100
print "%2f"%(Recall)
db.close()
```

Tabla 15 Script para obtener el Recall de Imphash

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"" Author: Richard Rivera
    Script for parameters for malware Clustering
""
import MySQLdb as mdb
import sys

db = mdb.connect('localhost', 'user', 'password', 'RELEASE1.0')

cursorC = db.cursor()
cursorT= db.cursor()
cursorR=db.cursor()
cursorN=db.cursor()
#getting n from the valid samples. records with Pehash are the valid
samples
cursorN.execute("SELECT COUNT(FILE_ID) FROM FILES where PEHASH IS
NOT NULL and FAMILY IS NOT NULL")
(N,)=cursorN.fetchone()
print N
Recall=0.0
R=0.0
Rj=0.0
cursorC.execute("SELECT DISTINCT FAMILY FROM FILES where PEHASH IS
NOT NULL and FAMILY IS NOT NULL")
for rowC in cursorC.fetchall():
    cursorT.execute("SELECT DISTINCT PEHASH FROM FILES where PEHASH
IS NOT NULL and FAMILY IS NOT NULL")
    maxJ=0
    for rowT in cursorT.fetchall():
        cursorR.execute("SELECT COUNT(PEHASH) FROM FILES WHERE
PEHASH='%s'AND FAMILY='%s'"%(rowT[0],rowC[0] ))
        (cR,)=cursorR.fetchone()
        if maxJ<cR:
            maxJ=cR
    Rj=Rj+maxJ
print cursorC.rowcount
print Rj
Recall=float(Rj)/N*100
print "%2f"%(Recall)
db.close()
```

Tabla 16 Script para obtener el Recall de Pehash

Estos 4 scripts implementan las métricas presentadas en el apartado 4.3 para las características estáticas de ficheros ejecutables Imphash y Pehash, estas métricas requieren de dos Clustering el de referencia T y el clustering a evaluar C, en los Scripts estos clustering están representados por los cursores de Python que traen el identificador de cada clúster para T y C desde la base de datos y para el caso de T es el

clustering actual de MALICIA el que tiene realizada la Clasificación por Familias de Malware es decir el Clustering de Referencia y el clustering C son todos los clústers que se generan de agrupar los ficheros PE que comparten el mismo valor de Imphash o Pehash según cada script.

El resultado de la ejecución de los scripts anteriores son las métricas Precision y Recall para Imphash y Pehash que se presentan en la siguiente tabla:

Característica	Precision	Recall	F-Measure
Imphash	**99,76**	**5,69**	**10,77**
Pehash	**99,99**	**8,05**	**14,90**

Tabla 17 Precision y Recall de Imphash y Pehash

Los resultados de la Tabla 17 para las dos características estáticas de ficheros ejecutables Imphash y Pehash nos presentan valores altos para la métrica Precision y valores bajos para la métrica Recall, esto indica que los clústers C corresponden casi en su totalidad a clústers de T, pero que por cada clúster de T hay varios clústers de C.

4.4.4. Clasificación de MALICIA con las características estáticas

Una vez que se obtuvo las métricas de Precision y Recall para las características estáticas Pehash e Imphash con un alto índice para Precision era factible la clasificación y a continuación se describe el proceso realizado para esto.

En el apartado 2.3.2 se revisó brevemente las técnicas de clasificación, para este trabajo se utilizará un método utilizado en el artículo de (Rafique & Caballero, 2013), denominado Clustering agresivo

Clustering agresivo. Esta agrupación agresiva calcula primero una métrica de similitud booleana entre dos ficheros ejecutables PE, o para referirnos a la clasificación de los clústers esta similitud se puede interpretar como la distancia entre las características de los elementos a clasificar. Los ficheros ejecutables tienen dos características a analizar el Imphash y el Pehash, la métrica de similitud booleana será 0 cuando dos ficheros ejecutables PE comparten una de estas dos características (OR lógico) y será 1 cuando no compartan ninguna característica. Entonces, se iterará sobre los ficheros ejecutables y se comprueba si el fichero ejecutable PE actual es similar a cualquiera de los ficheros ejecutables PE que ya están en un clúster. Si el fichero ejecutable PE actual es sólo similar a los ficheros ejecutables PE en el mismo grupo, añadimos el fichero ejecutable PE a ese clúster. Si es similar a los ficheros ejecutables PE en varios clústeres, se fusionan esos clústers y se añade el fichero ejecutable PE actual al clúster fusionado. Si no es similar a ningún fichero ejecutable PE en los clústers, se creará un nuevo clúster para éste.

Para aplicar este algoritmo de clasificación, se separó los ficheros ejecutables PE válidos contando los que actualmente no tienen asignada una familia y se los coloco en una nueva tabla de base de datos temporal llamada FILE_TMPC, con únicamente los atributos necesarios para realizar la clasificación. Se selecciona también el atributo ICON_LABEL que es una característica estática de los ficheros ejecutables PE que ya se

encuentra integrada en la clasificación actual de MALICIA, está se utilizará más adelante. Los atributos de esta tabla nueva tabla FILE_TMPC son los siguientes

- FILE_ID
- FAMILY
- IMPHASH
- PEHASH
- ICON_LABEL
- CLUSTER1
- CLUSTER2
- FAMILY1
- FAMILY2

Al clasificar los elementos de acuerdo al algoritmo mencionado clustering agresivo los clúster nuevos se irán colocando en el atributo CLUSTER1, para realizar esto se codificó otro script en Python llamado clusteringPehashImphash.py con este algoritmo, el cual se presenta en la siguiente tabla.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"" Author: Richard Rivera
    Script for malware Clustering (Pehash, Imphash)
""
import MySQLdb as mdb
import sys
db = mdb.connect('localhost', 'user', 'password', 'RELEASE1.0')
cursori = db.cursor()
cursorj= db.cursor()
cursorUp1=db.cursor()
cursorUp2=db.cursor()
cursorS=db.cursor()
clusterNumber=1
cursori.execute("SELECT IMPHASH, PEHASH, FILE_ID FROM FILE_TMPC
ORDER BY FILE_ID")
for rowi in cursori.fetchall():
    cursorS.execute("SELECT CLUSTER1 FROM FILE_TMPC WHERE
FILE_ID={}".format(rowi[2]))
    (S,)=cursorS.fetchone()
    if (S==None):
        cursorUp1.execute("UPDATE FILE_TMPC SET CLUSTER ='cluster{}'
WHERE FILE_ID='{}'".format(clusterNumber,rowi[2]))
        db.commit()
        cursorj.execute("SELECT IMPHASH, PEHASH, FILE_ID FROM
FILE_TMPC ORDER BY FILE_ID")
        for rowj in cursorj.fetchall():
            cursorS.execute("SELECT CLUSTER1 FROM FILE_TMPC WHERE
FILE_ID={}".format(rowj[2]))
            (S,)=cursorS.fetchone()
            if (rowi[2]<rowj[2] and S==None):
                if (rowi[0]==rowj[0] or (rowi[1]==rowj[1] and
rowi[1]!=None)):
                    cursorUp2.execute("UPDATE FILE_TMPC SET CLUSTER1
='cluster{}' WHERE FILE_ID='{}'".format(clusterNumber,rowj[2]))
                    db.commit()
        clusterNumber=clusterNumber+1
db.close()
```

Tabla 18 Script Clustering Agresivo con Pehash e Imphash

El resultado del script de la taba anterior genera 1136 nuevos clústers agrupados con el algoritmo clustering agresivo los cuales se presentan en la siguiente tabla:

Cluster_ID	#
cluster1898	363
cluster3582	327
cluster1081	285
cluster4622	227
cluster7633	206
cluster3839	203
cluster8521	190
cluster8151	189
cluster6628	160
cluster3019	144
cluster4950	139
cluster6905	138
cluster7455	123
cluster7849	123
cluster709	121
cluster4142	114
cluster3347	105
cluster2237	105
cluster7157	100
cluster1432	99
cluster8020	99
cluster6417	91
cluster3680	89
cluster3341	83
cluster5536	83
cluster195	83
cluster8568	82
cluster8942	82
cluster8865	75
cluster6664	73
cluster4707	69
cluster10446	68
cluster3959	64
cluster7275	62
cluster3543	62
cluster998	60
cluster2273	57
cluster5402	54
cluster9195	49
cluster7314	49
cluster4573	48
cluster2041	47
cluster6076	47
cluster8491	47
cluster5868	47
cluster1946	46
cluster5193	46
cluster8353	46
cluster10187	46
cluster526	45
cluster9889	45
cluster5257	45
cluster309	45
cluster9987	44
cluster4232	44
cluster6867	43
cluster3032	43
cluster5727	42
cluster18	42
cluster1996	42
cluster3851	41
cluster5819	41
cluster6276	40
cluster7848	40
cluster2909	39
cluster546	39
cluster421	39
cluster1774	39
cluster6536	38
cluster11121	38
cluster2494	37
cluster9786	36
cluster1691	36
cluster736	36
cluster8439	36
cluster9785	35
cluster1050	35
cluster4510	35
cluster1148	35
cluster10037	35
cluster7584	35
cluster8814	35
cluster8481	34
cluster3023	33
cluster10987	33
cluster10232	33
cluster11074	33
cluster10390	33
cluster1426	32
cluster5979	32
cluster10410	32
cluster1830	32
cluster9660	31
cluster4642	31
cluster7420	31
cluster365	30
cluster10779	30
cluster955	30
cluster9159	30
cluster9291	30
cluster5004	30
cluster5331	29
cluster2117	29
cluster2128	29
cluster14	28
cluster6289	27
cluster10538	27
cluster6647	27
cluster5720	27
cluster9098	26
cluster4849	26
cluster749	26
cluster10825	26
cluster11040	26
cluster1913	25
cluster8014	25
cluster66	25
cluster1964	25
cluster10365	24
cluster1839	24
cluster2190	24
cluster5363	24
cluster10278	24
cluster3691	24
cluster485	24
cluster10938	23
cluster10915	23
cluster1695	23
cluster6016	23
cluster10112	22
cluster10887	22
cluster3151	22
cluster6220	22
cluster9216	22
cluster10753	22
cluster2340	22
cluster6397	21
cluster10715	21
cluster2374	21
cluster3349	21
cluster2013	21
cluster6195	21
cluster2150	20
cluster5962	20
cluster5624	20
cluster640	20
cluster11177	20
cluster653	20
cluster5516	19
cluster9479	19
cluster8406	19
cluster9275	19
cluster2879	19
cluster242	19
cluster9434	19
cluster10147	19
cluster10631	18
cluster11234	18
cluster994	18
cluster8421	18
cluster5791	18
cluster6352	18
cluster11254	18
cluster2855	18
cluster10690	17
cluster6154	17
cluster6943	17
cluster3895	17
cluster2780	17
cluster6489	17
cluster7278	17
cluster1542	17
cluster201	17
cluster3279	16
cluster10344	16
cluster1680	16
cluster10968	16
cluster5475	16
cluster6493	15
cluster1624	15
cluster1754	15
cluster10583	15
cluster6277	15
cluster1052	15
cluster8048	15
cluster9535	15
cluster7063	15
cluster6063	15
cluster10865	15
cluster8850	14
cluster10309	14

cluster1745	14
cluster2465	14
cluster6455	14
cluster9405	14
cluster6258	14
cluster480	14
cluster11213	14
cluster1024	14
cluster5863	13
cluster9379	13
cluster836	13
cluster5295	13
cluster11320	13
cluster6173	13
cluster5462	13
cluster9140	13
cluster1657	12
cluster9510	12
cluster7147	12
cluster534	12
cluster11307	12
cluster9371	12
cluster11028	11
cluster5809	11
cluster10084	11
cluster9063	11
cluster6845	11
cluster11166	11
cluster5460	11
cluster2286	11
cluster7407	11
cluster568	11
cluster362	11
cluster6411	11
cluster9349	10
cluster10604	10
cluster5887	10
cluster10526	10
cluster6008	10
cluster900	10
cluster9875	10
cluster7597	9
cluster2502	9
cluster5691	9
cluster9066	9
cluster10168	9
cluster175	9
cluster3215	9
cluster9975	9
cluster9419	9
cluster993	9
cluster5322	9

cluster185	9
cluster109	8
cluster6227	8
cluster137	8
cluster3241	8
cluster5695	8
cluster5928	8
cluster10139	8
cluster11287	8
cluster5306	8
cluster6092	8
cluster6303	8
cluster11340	8
cluster8	8
cluster10574	8
cluster5394	8
cluster9495	8
cluster6460	8
cluster10817	7
cluster5012	7
cluster855	7
cluster9326	7
cluster6188	7
cluster6403	7
cluster9951	7
cluster2848	7
cluster1952	7
cluster5960	7
cluster9462	7
cluster9629	7
cluster400	7
cluster11197	7
cluster504	7
cluster10327	7
cluster2456	6
cluster3298	6
cluster11280	6
cluster5943	6
cluster2244	6
cluster6644	6
cluster1697	6
cluster1968	6
cluster3310	6
cluster10611	6
cluster11206	6
cluster3172	6
cluster9645	6
cluster6441	6
cluster4487	6
cluster7347	6
cluster579	6
cluster10106	6

cluster9277	6
cluster10683	6
cluster3263	6
cluster6094	6
cluster10736	6
cluster6288	5
cluster7419	5
cluster903	5
cluster8329	5
cluster7916	5
cluster10818	5
cluster1570	5
cluster11089	5
cluster10632	5
cluster529	5
cluster9390	5
cluster9944	5
cluster399	5
cluster5250	5
cluster3002	5
cluster6669	5
cluster6052	5
cluster3051	5
cluster5654	5
cluster6396	5
cluster11229	5
cluster9030	5
cluster5939	5
cluster5619	5
cluster10355	5
cluster9895	4
cluster1682	4
cluster9283	4
cluster3043	4
cluster10666	4
cluster1545	4
cluster10099	4
cluster120	4
cluster10469	4
cluster7402	4
cluster9885	4
cluster2756	4
cluster9751	4
cluster10879	4
cluster2611	4
cluster6434	4
cluster10678	4
cluster11302	4
cluster9618	4
cluster10135	4
cluster361	4
cluster10433	4

cluster1684	4
cluster2624	4
cluster10219	4
cluster3047	4
cluster9964	4
cluster11070	4
cluster10645	4
cluster9873	4
cluster10258	4
cluster11336	4
cluster2616	4
cluster187	4
cluster10573	4
cluster139	4
cluster115	4
cluster10401	4
cluster2048	4
cluster7190	4
cluster10855	4
cluster9172	4
cluster2730	4
cluster5371	3
cluster6108	3
cluster9617	3
cluster1874	3
cluster9603	3
cluster10610	3
cluster2040	3
cluster9584	3
cluster10167	3
cluster172	3
cluster1910	3
cluster6392	3
cluster7533	3
cluster9611	3
cluster11259	3
cluster3171	3
cluster9656	3
cluster599	3
cluster9590	3
cluster3106	3
cluster9817	3
cluster491	3
cluster1879	3
cluster9768	3
cluster9879	3
cluster10667	3
cluster3011	3
cluster9776	3
cluster10993	3
cluster10408	3
cluster11220	3

cluster2763	3
cluster6234	3
cluster9336	3
cluster10682	3
cluster1925	3
cluster3333	3
cluster9477	3
cluster11247	3
cluster2203	3
cluster6557	3
cluster10055	3
cluster279	3
cluster10618	3
cluster10735	3
cluster2602	3
cluster9965	3
cluster10173	3
cluster1407	3
cluster9891	3
cluster10630	3
cluster3335	3
cluster9955	3
cluster169	3
cluster10532	3
cluster118	3
cluster635	3
cluster945	3
cluster6222	3
cluster9306	3
cluster10676	3
cluster11300	2
cluster5878	2
cluster8405	2
cluster9142	2
cluster9402	2
cluster9779	2
cluster135	2
cluster10765	2
cluster1421	2
cluster628	2
cluster6255	2
cluster9460	2
cluster9718	2
cluster92	2
cluster271	2
cluster1041	2
cluster11204	2
cluster2622	2
cluster3593	2
cluster9855	2
cluster9976	2
cluster6412	2

cluster8414	2
cluster10342	2
cluster9774	2
cluster10747	2
cluster2863	2
cluster9464	2
cluster99	2
cluster1087	2
cluster7226	2
cluster11331	2
cluster5959	2
cluster5	2
cluster7598	2
cluster984	2
cluster8177	2
cluster9605	2
cluster9727	2
cluster9897	2
cluster10087	2
cluster111	2
cluster5098	2
cluster9720	2
cluster94	2
cluster876	2
cluster10892	2
cluster5492	2
cluster5710	2
cluster6217	2
cluster9748	2
cluster772	2
cluster2506	2
cluster7097	2
cluster722	2
cluster1557	2
cluster3292	2
cluster6103	2
cluster9613	2
cluster10105	2
cluster124	2
cluster1401	2
cluster8809	2
cluster9089	2
cluster9724	2
cluster101	2
cluster10628	2
cluster10944	2
cluster11354	2
cluster6473	2
cluster64	2
cluster844	2
cluster10884	2
cluster9028	2

cluster9530	2
cluster9832	2
cluster503	2
cluster1596	2
cluster9954	2
cluster10655	2
cluster10782	2
cluster8258	2
cluster9122	2
cluster9566	2
cluster9902	2
cluster631	2
cluster10635	2
cluster1159	2
cluster5100	2
cluster9557	2
cluster9759	2
cluster96	2
cluster5493	2
cluster6457	2
cluster9869	2
cluster10011	2
cluster11	2
cluster10697	2
cluster11137	2
cluster7110	2
cluster9792	2
cluster133	2
cluster10336	2
cluster9601	2
cluster9658	2
cluster9765	2
cluster104	2
cluster10308	2
cluster610	2
cluster9554	2
cluster572	2
cluster2620	2
cluster8409	2
cluster9623	2
cluster9784	2
cluster690	2
cluster5856	2
cluster9478	2
cluster9903	2
cluster10974	2
cluster3169	2
cluster9598	2
cluster9655	2
cluster9883	2
cluster293	2
cluster883	2

cluster7192	2
cluster9750	2
cluster835	2
cluster5415	2
cluster6185	2
cluster6433	2
cluster9025	2
cluster9526	2
cluster9579	2
cluster10562	2
cluster741	2
cluster1562	1
cluster11084	1
cluster1916	1
cluster5646	1
cluster6904	1
cluster9011	1
cluster9486	1
cluster9572	1
cluster9700	1
cluster9733	1
cluster9950	1
cluster10338	1
cluster417	1
cluster10504	1
cluster672	1
cluster10649	1
cluster970	1
cluster2278	1
cluster3221	1
cluster4313	1
cluster5280	1
cluster6089	1
cluster6661	1
cluster8176	1
cluster9365	1
cluster9469	1
cluster9564	1
cluster9726	1
cluster9766	1
cluster360	1
cluster10414	1
cluster1155	1
cluster10959	1
cluster11357	1
cluster2798	1
cluster3139	1
cluster5509	1
cluster5718	1
cluster6045	1
cluster7280	1
cluster9555	1

cluster9595	1
cluster9649	1
cluster9756	1
cluster9878	1
cluster10045	1
cluster10262	1
cluster10398	1
cluster577	1
cluster10727	1
cluster11341	1
cluster3117	1
cluster5488	1
cluster5709	1
cluster6215	1
cluster6453	1
cluster8034	1
cluster8485	1
cluster9034	1
cluster9439	1
cluster9533	1
cluster9637	1
cluster9711	1
cluster9747	1
cluster9	1
cluster10210	1
cluster190	1
cluster10582	1
cluster759	1
cluster10685	1
cluster10833	1
cluster11119	1
cluster11313	1
cluster5650	1
cluster6161	1
cluster7073	1
cluster7735	1
cluster9019	1
cluster9420	1
cluster9499	1
cluster9576	1
cluster9627	1
cluster9704	1
cluster9740	1
cluster9963	1
cluster1	1
cluster428	1
cluster10536	1
cluster10791	1
cluster11069	1
cluster11293	1
cluster2341	1
cluster3008	1

cluster4500	1
cluster6099	1
cluster9129	1
cluster9569	1
cluster9674	1
cluster9730	1
cluster9918	1
cluster10328	1
cluster10644	1
cluster954	1
cluster1273	1
cluster5513	1
cluster6073	1
cluster8168	1
cluster8787	1
cluster9086	1
cluster9362	1
cluster9559	1
cluster9599	1
cluster9723	1
cluster9762	1
cluster10073	1
cluster10299	1
cluster295	1
cluster10407	1
cluster10627	1
cluster893	1
cluster11217	1
cluster2108	1
cluster11353	1
cluster3127	1
cluster3710	1
cluster5506	1
cluster5715	1
cluster6018	1
cluster6461	1
cluster8061	1
cluster9043	1
cluster9457	1
cluster9552	1
cluster9644	1
cluster9715	1
cluster9872	1
cluster10023	1
cluster221	1
cluster10394	1
cluster1030	1
cluster1677	1
cluster3492	1
cluster4696	1
cluster7130	1
cluster8446	1

cluster9026	1
cluster9233	1
cluster9432	1
cluster9529	1
cluster9580	1
cluster9632	1
cluster9708	1
cluster9744	1
cluster9967	1
cluster10177	1
cluster186	1
cluster10356	1
cluster10572	1
cluster1924	1
cluster3029	1
cluster4585	1
cluster5378	1
cluster5647	1
cluster6111	1
cluster9012	1
cluster9151	1
cluster9404	1
cluster9487	1
cluster9573	1
cluster9701	1
cluster9734	1
cluster9782	1
cluster10339	1
cluster418	1
cluster10510	1
cluster681	1
cluster10651	1
cluster11038	1
cluster11282	1
cluster2935	1
cluster4344	1
cluster5524	1
cluster6091	1
cluster8864	1
cluster9114	1
cluster9368	1
cluster9470	1
cluster9565	1
cluster9666	1
cluster10323	1
cluster629	1
cluster909	1
cluster1158	1
cluster10964	1
cluster11242	1
cluster11358	1
cluster3143	1

cluster3867	1
cluster5510	1
cluster5719	1
cluster6046	1
cluster6257	1
cluster8154	1
cluster8580	1
cluster9350	1
cluster9461	1
cluster9556	1
cluster9596	1
cluster9651	1
cluster9757	1
cluster10052	1
cluster10272	1
cluster278	1
cluster10399	1
cluster578	1
cluster10730	1
cluster11350	1
cluster3122	1
cluster3620	1
cluster4904	1
cluster7163	1
cluster8035	1
cluster8486	1
cluster9035	1
cluster9290	1
cluster9450	1
cluster9534	1
cluster9587	1
cluster9639	1
cluster9712	1
cluster9865	1
cluster10	1
cluster191	1
cluster10367	1
cluster10834	1
cluster4659	1
cluster5405	1
cluster5653	1
cluster5945	1
cluster8418	1
cluster9022	1
cluster9203	1
cluster9427	1
cluster9577	1
cluster9628	1
cluster9705	1
cluster9741	1
cluster2	1
cluster433	1

cluster10804	1
cluster11296	1
cluster5644	1
cluster5866	1
cluster6394	1
cluster7551	1
cluster8948	1
cluster9138	1
cluster9484	1
cluster9570	1
cluster9698	1
cluster9731	1
cluster10335	1
cluster10482	1
cluster648	1
cluster10751	1
cluster1857	1
cluster11278	1
cluster2878	1
cluster4147	1
cluster5514	1
cluster5728	1
cluster8174	1
cluster9363	1
cluster9465	1
cluster9560	1
cluster9600	1
cluster9657	1
cluster9763	1
cluster10074	1
cluster10302	1
cluster305	1
cluster607	1
cluster894	1
cluster1711	1
cluster3129	1
cluster3769	1
cluster4959	1
cluster5507	1
cluster5716	1
cluster6022	1
cluster8072	1
cluster8524	1
cluster9045	1
cluster9458	1
cluster9553	1
cluster9593	1
cluster9716	1
cluster9754	1
cluster222	1
cluster10396	1
cluster10606	1

cluster10717	1
cluster1031	1
cluster3112	1
cluster6191	1
cluster9433	1
cluster9581	1
cluster9634	1
cluster9709	1
cluster9745	1
cluster9974	1
cluster6	1
cluster10184	1
cluster10358	1
cluster754	1
cluster11105	1
cluster11303	1
cluster2471	1
cluster3030	1
cluster4595	1
cluster5381	1
cluster5648	1
cluster6410	1
cluster8407	1
cluster9016	1
cluster9490	1
cluster9574	1
cluster9622	1
cluster9702	1
cluster9737	1
cluster9783	1
cluster10340	1
cluster683	1
cluster989	1
cluster1891	1
cluster2326	1
cluster3261	1
cluster5854	1
cluster9369	1
cluster9608	1
cluster9669	1
cluster9728	1
cluster9772	1
cluster10088	1
cluster10326	1
cluster927	1
cluster2853	1
cluster5511	1
cluster8163	1
cluster8590	1
cluster9073	1
cluster9352	1
cluster9597	1

cluster9654	1
cluster9721	1
cluster9882	1
cluster882	1
cluster10910	1
cluster11208	1
cluster11351	1
cluster2628	1
cluster3124	1
cluster4905	1
cluster5712	1
cluster9037	1
cluster9453	1
cluster9588	1
cluster9641	1
cluster9713	1
cluster9749	1
cluster10223	1
cluster10383	1
cluster10598	1
cluster822	1
cluster1668	1
cluster1963	1
cluster11324	1
cluster4694	1
cluster5410	1
cluster5957	1
cluster6175	1
cluster6419	1
cluster9024	1
cluster9430	1
cluster9515	1
cluster9578	1
cluster9706	1
cluster9742	1
cluster3	1
cluster184	1
cluster10352	1
cluster10540	1
cluster10669	1
cluster1005	1
cluster10810	1
cluster1561	1
cluster1914	1
cluster11299	1
cluster2385	1
cluster3293	1
cluster4514	1
cluster5365	1
cluster5645	1
cluster6107	1
cluster6873	1

cluster8378	1
cluster9010	1
cluster9394	1
cluster9485	1
cluster9571	1
cluster9616	1
cluster9699	1
cluster9732	1
cluster9778	1
cluster9948	1
cluster10497	1
cluster10646	1
cluster958	1
cluster11022	1
cluster1865	1
cluster11279	1
cluster2277	1
cluster5515	1
cluster6079	1
cluster6280	1
cluster6652	1
cluster7410	1
cluster8175	1
cluster9095	1
cluster9364	1
cluster9468	1
cluster9562	1
cluster9725	1
cluster10082	1
cluster1152	1
cluster10952	1
cluster1739	1
cluster11356	1
cluster3135	1
cluster5508	1
cluster5717	1
cluster6023	1
cluster6242	1
cluster8076	1
cluster8532	1
cluster9046	1
cluster9344	1
cluster9459	1
cluster9594	1
cluster9648	1
cluster9717	1
cluster9755	1
cluster10043	1
cluster10260	1
cluster10397	1
cluster10607	1
cluster846	1

cluster10724	1	cluster9739	1	cluster8610	1	cluster535	1
cluster1032	1	cluster10341	1	cluster9081	1	cluster10602	1
cluster10885	1	cluster427	1	cluster9354	1	cluster10711	1
cluster1681	1	cluster10659	1	cluster9463	1	cluster1676	1
cluster3114	1	cluster10789	1	cluster9558	1	cluster11143	1
cluster4777	1	cluster11061	1	cluster9722	1	cluster11327	1
cluster5701	1	cluster11292	1	cluster9760	1	cluster2604	1
cluster5961	1	cluster3005	1	cluster10071	1	cluster3084	1
cluster6444	1	cluster4497	1	cluster98	1	cluster3411	1
cluster7154	1	cluster6673	1	cluster10291	1	cluster4695	1
cluster8484	1	cluster7457	1	cluster10403	1	cluster5666	1
cluster9532	1	cluster8287	1	cluster584	1	cluster5958	1
cluster9582	1	cluster8941	1	cluster10626	1	cluster7124	1
cluster9636	1	cluster9126	1	cluster2064	1	cluster7858	1
cluster9710	1	cluster9567	1	cluster11352	1	cluster9232	1
cluster9746	1	cluster9610	1	cluster3125	1	cluster9431	1
cluster9853	1	cluster9670	1	cluster4906	1	cluster9630	1
cluster188	1	cluster9729	1	cluster5495	1	cluster9707	1
cluster10360	1	cluster9773	1	cluster5713	1	cluster9743	1
cluster755	1	cluster10096	1	cluster8050	1	cluster9815	1
cluster10832	1	cluster363	1	cluster8500	1	cluster9966	1
cluster1608	1	cluster10452	1	cluster9042	1	cluster4	1
cluster11106	1	cluster10637	1	cluster9455	1	cluster10174	1
cluster1942	1	cluster1209	1	cluster9550	1		
cluster5649	1	cluster5165	1	cluster9589	1		
cluster6160	1	cluster5512	1	cluster9643	1		
cluster7732	1	cluster5723	1	cluster9714	1		
cluster9017	1	cluster6275	1	cluster9871	1		
cluster9575	1	cluster7382	1	cluster10016	1		
cluster9703	1	cluster8167	1	cluster10228	1		

Tabla 19 Clústers Pehash-Imphash

Como se presentó en el apartado 3.1 la clasificación actual de MALICIA tiene 53 familias, con nuestro algoritmo hemos obtenido 1136 familias, dato que no es muy alentador, pero tomando en cuenta los resultados de la Tabla 8, antes se tenía 1500 clústers de Pehash y 2028 de Imphash, y ahora al agrupar estas dos características estáticas de los PE hemos reducido este valor a 1136 clústers.

Para mejorar la clasificación se probará una vez más con el mismo algoritmo clustering agresivo, pero esta vez añadiendo el atributo ICON_LABEL (Nappa, Rafique, & Caballero, 2013), es decir en este caso la distancia entre los elementos a comparar la métrica de similitud será 0 si comparten por lo menos uno (OR Lógico) de los tres atributos Pehash, Imphash e ICON_LABEL; y será 1 si no comparten ninguno de los atributos. Luego el proceso de creación de clústers será la misma que en el caso anterior.

Para esta nueva clasificación se creó un script similar al anterior con las respectivas modificaciones, este se presenta en la siguiente tabla, el valor generado de los nuevos clústers se almacenarán en el atributo CLUSTER2 de la tabla FILE_TMPC.

```
 #!/usr/bin/python
# -*- coding: utf-8 -*-
"" Author: Richard Rivera
    Script for malware Clustering ( Pehash, Imphash, icon_label)
""
```

```
import MySQLdb as mdb
import sys
db = mdb.connect('localhost', 'user', 'password', 'RELEASE1.0')
cursori = db.cursor()
cursorj= db.cursor()
cursorUpi=db.cursor()
cursorUpj=db.cursor()
cursorSelecti=db.cursor()
cursorSelectj=db.cursor()
clusterNumber=1
cursori.execute("SELECT FILE_ID, PEHASH, IMPHASH, ICON_LABEL FROM
FILE_TMPC ORDER BY FILE_ID")
for rowi in cursori.fetchall():
    cursorSelecti.execute("SELECT CLUSTER2 FROM FILE_TMPC WHERE
FILE_ID={}".format(rowi[0]))
    (Selecti,)=cursorSelecti.fetchone()
    if (Selecti==None):
        Selecti=clusterNumber
        cursorUpi.execute("UPDATE FILE_TMPC SET CLUSTER2 ='{}' WHERE
FILE_ID='{}'".format(Selecti,rowi[0]))
        db.commit()
        clusterNumber=clusterNumber+1
    cursorj.execute("SELECT FILE_ID, PEHASH, IMPHASH, ICON_LABEL
FROM FILE_TMPC ORDER BY FILE_ID")
    for rowj in cursorj.fetchall():
        cursorSelectj.execute("SELECT CLUSTER2 FROM FILE_TMPC WHERE
FILE_ID={}".format(rowj[0]))
        (Selectj,)=cursorSelectj.fetchone()
        if ((rowi[0]<rowj[0] and Selectj==None)and
(rowi[1]==rowj[1] or (rowi[2]==rowj[2] and rowi[2]!=None) or
(rowi[3]==rowj[3] and rowi[3]!=None))):
            cursorUpj.execute("UPDATE FILE_TMPC SET CLUSTER2 ='{}'
WHERE FILE_ID='{}'".format(Selecti,rowj[0]))
            db.commit()
db.close()
```

Tabla 20 Script Clustering Agresivo con Pehash, Imphash e ICON_LABEL

El resultado del script de la taba anterior genera 858 nuevos clústers agrupados con el algoritmo clustering agresivo los cuales se presentan en la siguiente tabla:

Cluster_ID	#
cluster5	3238
cluster274	1074
cluster96	650
cluster134	189
cluster49	184
cluster273	160
cluster16	150
cluster354	130
cluster195	124
cluster329	117
cluster19	100
cluster319	94
cluster12	92
cluster258	91
cluster309	91
cluster62	82

cluster294	77
cluster26	73
cluster706	68
cluster146	66
cluster278	59
cluster315	56
cluster357	55
cluster43	50
cluster301	49
cluster194	47
cluster78	47
cluster352	47
cluster79	46
cluster659	46
cluster611	45
cluster630	44
cluster126	44

cluster101	43
cluster298	43
cluster186	42
cluster189	41
cluster400	41
cluster318	40
cluster244	40
cluster812	38
cluster216	37
cluster88	37
cluster282	36
cluster593	36
cluster67	36
cluster634	35
cluster592	35
cluster797	33
cluster694	33

cluster806	33
cluster664	33
cluster99	33
cluster366	33
cluster58	32
cluster703	32
cluster308	31
cluster523	31
cluster769	30
cluster205	30
cluster398	30
cluster153	29
cluster363	28
cluster275	27
cluster716	27
cluster184	27
cluster248	27

cluster346	26
cluster389	26
cluster777	26
cluster802	26
cluster321	25
cluster7	25
cluster669	24
cluster691	24
cluster81	24
cluster409	24
cluster25	24
cluster790	23
cluster207	23
cluster789	23
cluster115	22
cluster649	22
cluster84	22
cluster767	22
cluster786	22
cluster758	21
cluster86	21
cluster76	21
cluster254	21
cluster120	21
cluster204	20
cluster816	20
cluster80	20
cluster402	20
cluster652	19
cluster95	19
cluster341	19
cluster405	19
cluster250	18
cluster176	18
cluster187	18
cluster828	18
cluster736	18
cluster825	18
cluster285	17
cluster755	17
cluster92	17
cluster299	17
cluster119	16
cluster226	16
cluster795	16
cluster234	16
cluster686	16
cluster61	15
cluster214	15
cluster271	15
cluster723	15
cluster286	15
cluster245	15
cluster782	15
cluster32	15
cluster324	15
cluster424	14
cluster265	14
cluster674	14
cluster821	14
cluster270	14
cluster242	14
cluster844	13
cluster102	13
cluster145	13
cluster192	13
cluster448	13
cluster229	12
cluster842	12
cluster395	12
cluster292	12
cluster54	11
cluster815	11
cluster83	11
cluster152	11
cluster305	11
cluster643	11
cluster188	11
cluster800	11
cluster413	10
cluster605	10
cluster726	10
cluster713	10
cluster206	10
cluster22	9
cluster628	9
cluster35	9
cluster348	9
cluster654	9
cluster281	9
cluster481	8
cluster453	8
cluster834	8
cluster383	8
cluster721	8
cluster849	8
cluster220	8
cluster117	8
cluster455	8
cluster249	8
cluster267	8
cluster651	8
cluster364	7
cluster677	7
cluster425	7
cluster621	7
cluster255	7
cluster817	7
cluster202	7
cluster74	7
cluster775	7
cluster505	7
cluster107	7
cluster221	6
cluster648	6
cluster730	6
cluster819	6
cluster197	6
cluster753	6
cluster764	6
cluster832	6
cluster253	5
cluster406	5
cluster232	5
cluster776	5
cluster808	5
cluster737	5
cluster378	5
cluster307	5
cluster247	5
cluster279	5
cluster262	5
cluster824	5
cluster618	5
cluster213	5
cluster313	5
cluster196	5
cluster399	4
cluster751	4
cluster783	4
cluster296	4
cluster840	4
cluster705	4
cluster90	4
cluster261	4
cluster613	4
cluster741	4
cluster805	4
cluster407	4
cluster439	4
cluster55	4
cluster304	4
cluster720	4
cluster848	4
cluster610	4
cluster699	4
cluster500	4
cluster781	4
cluster646	4
cluster497	4
cluster650	4
cluster604	4
cluster661	4
cluster302	4
cluster103	4
cluster576	4
cluster665	4
cluster747	4
cluster708	4
cluster829	3
cluster502	3
cluster655	3
cluster488	3
cluster520	3
cluster225	3
cluster410	3
cluster499	3
cluster492	3
cluster748	3
cluster446	3
cluster638	3
cluster702	3
cluster798	3
cluster311	3
cluster567	3
cluster823	3
cluster272	3
cluster624	3
cluster752	3
cluster201	3
cluster386	3
cluster251	3
cluster411	3
cluster731	3
cluster763	3
cluster827	3
cluster91	3
cluster596	3
cluster237	3
cluster589	3
cluster653	3
cluster77	3
cluster607	3
cluster735	3
cluster586	3
cluster714	3
cluster579	3
cluster238	3

cluster750	3
cluster729	3
cluster612	3
cluster765	2
cluster335	2
cluster591	2
cluster847	2
cluster584	2
cluster616	2
cluster680	2
cluster257	2
cluster289	2
cluster609	2
cluster673	2
cluster65	2
cluster314	2
cluster467	2
cluster787	2
cluster236	2
cluster396	2
cluster556	2
cluster44	2
cluster485	2
cluster549	2
cluster37	2
cluster574	2
cluster734	2
cluster151	2
cluster343	2
cluster375	2
cluster599	2
cluster631	2
cluster791	2
cluster855	2
cluster240	2
cluster336	2
cluster464	2
cluster784	2
cluster297	2
cluster361	2
cluster457	2
cluster553	2
cluster745	2
cluster738	2
cluster770	2
cluster756	2
cluster269	2
cluster685	2
cluster813	2
cluster614	2
cluster191	2
cluster223	2

cluster344	2
cluster177	2
cluster266	2
cluster362	2
cluster522	2
cluster387	2
cluster156	2
cluster796	2
cluster437	2
cluster597	2
cluster629	2
cluster21	2
cluster85	2
cluster622	2
cluster718	2
cluster231	2
cluster391	2
cluster423	2
cluster519	2
cluster615	2
cluster647	2
cluster839	2
cluster288	2
cluster768	2
cluster377	2
cluster441	2
cluster473	2
cluster601	2
cluster57	2
cluster338	2
cluster594	2
cluster818	2
cluster459	2
cluster260	2
cluster548	2
cluster644	2
cluster157	1
cluster317	1
cluster349	1
cluster381	1
cluster445	1
cluster477	1
cluster509	1
cluster541	1
cluster573	1
cluster637	1
cluster701	1
cluster733	1
cluster29	1
cluster93	1
cluster125	1
cluster150	1

cluster182	1
cluster246	1
cluster310	1
cluster342	1
cluster374	1
cluster438	1
cluster470	1
cluster534	1
cluster566	1
cluster598	1
cluster662	1
cluster822	1
cluster854	1
cluster118	1
cluster143	1
cluster175	1
cluster239	1
cluster303	1
cluster367	1
cluster431	1
cluster463	1
cluster495	1
cluster527	1
cluster559	1
cluster623	1
cluster687	1
cluster719	1
cluster15	1
cluster47	1
cluster111	1
cluster136	1
cluster168	1
cluster200	1
cluster264	1
cluster328	1
cluster360	1
cluster392	1
cluster456	1
cluster552	1
cluster712	1
cluster744	1
cluster8	1
cluster40	1
cluster72	1
cluster104	1
cluster161	1
cluster193	1
cluster353	1
cluster385	1
cluster417	1
cluster449	1
cluster513	1

cluster545	1
cluster577	1
cluster641	1
cluster801	1
cluster833	1
cluster1	1
cluster33	1
cluster97	1
cluster129	1
cluster154	1
cluster218	1
cluster442	1
cluster474	1
cluster506	1
cluster538	1
cluster570	1
cluster602	1
cluster666	1
cluster698	1
cluster762	1
cluster794	1
cluster826	1
cluster858	1
cluster122	1
cluster147	1
cluster179	1
cluster211	1
cluster243	1
cluster339	1
cluster371	1
cluster403	1
cluster435	1
cluster531	1
cluster563	1
cluster595	1
cluster627	1
cluster851	1
cluster51	1
cluster140	1
cluster172	1
cluster268	1
cluster300	1
cluster332	1
cluster428	1
cluster460	1
cluster524	1
cluster588	1
cluster620	1
cluster684	1
cluster780	1
cluster108	1
cluster133	1

cluster165	1
cluster293	1
cluster325	1
cluster421	1
cluster517	1
cluster581	1
cluster645	1
cluster709	1
cluster773	1
cluster837	1
cluster69	1
cluster158	1
cluster190	1
cluster222	1
cluster350	1
cluster382	1
cluster414	1
cluster478	1
cluster510	1
cluster542	1
cluster606	1
cluster670	1
cluster766	1
cluster830	1
cluster30	1
cluster94	1
cluster183	1
cluster215	1
cluster471	1
cluster503	1
cluster535	1
cluster663	1
cluster695	1
cluster727	1
cluster759	1
cluster23	1
cluster87	1
cluster144	1
cluster208	1
cluster368	1
cluster432	1
cluster496	1
cluster528	1
cluster560	1
cluster656	1
cluster688	1
cluster48	1
cluster112	1
cluster137	1
cluster169	1
cluster233	1
cluster393	1

cluster489	1
cluster521	1
cluster585	1
cluster617	1
cluster681	1
cluster809	1
cluster841	1
cluster9	1
cluster41	1
cluster73	1
cluster105	1
cluster162	1
cluster290	1
cluster322	1
cluster418	1
cluster450	1
cluster482	1
cluster514	1
cluster546	1
cluster578	1
cluster642	1
cluster2	1
cluster34	1
cluster66	1
cluster98	1
cluster130	1
cluster155	1
cluster219	1
cluster283	1
cluster347	1
cluster379	1
cluster443	1
cluster475	1
cluster507	1
cluster539	1
cluster571	1
cluster603	1
cluster635	1
cluster667	1
cluster27	1
cluster59	1
cluster123	1
cluster148	1
cluster180	1
cluster212	1
cluster276	1
cluster340	1
cluster372	1
cluster404	1
cluster436	1
cluster468	1
cluster532	1

cluster564	1
cluster660	1
cluster692	1
cluster724	1
cluster788	1
cluster820	1
cluster852	1
cluster20	1
cluster52	1
cluster116	1
cluster141	1
cluster173	1
cluster333	1
cluster365	1
cluster397	1
cluster429	1
cluster461	1
cluster493	1
cluster525	1
cluster557	1
cluster717	1
cluster749	1
cluster845	1
cluster13	1
cluster45	1
cluster109	1
cluster166	1
cluster198	1
cluster230	1
cluster326	1
cluster358	1
cluster390	1
cluster422	1
cluster454	1
cluster486	1
cluster518	1
cluster550	1
cluster582	1
cluster678	1
cluster710	1
cluster742	1
cluster774	1
cluster838	1
cluster6	1
cluster38	1
cluster70	1
cluster159	1
cluster287	1
cluster351	1
cluster415	1
cluster447	1
cluster479	1

cluster511	1
cluster543	1
cluster575	1
cluster639	1
cluster671	1
cluster799	1
cluster831	1
cluster31	1
cluster63	1
cluster127	1
cluster280	1
cluster312	1
cluster376	1
cluster408	1
cluster440	1
cluster472	1
cluster504	1
cluster536	1
cluster568	1
cluster600	1
cluster632	1
cluster696	1
cluster728	1
cluster760	1
cluster792	1
cluster856	1
cluster24	1
cluster56	1
cluster209	1
cluster241	1
cluster337	1
cluster369	1
cluster401	1
cluster433	1
cluster465	1
cluster529	1
cluster561	1
cluster625	1
cluster657	1
cluster689	1
cluster785	1
cluster17	1
cluster113	1
cluster138	1
cluster170	1
cluster330	1
cluster394	1
cluster426	1
cluster458	1
cluster490	1
cluster554	1
cluster682	1

cluster746	1
cluster778	1
cluster810	1
cluster10	1
cluster42	1
cluster106	1
cluster163	1
cluster227	1
cluster259	1
cluster291	1
cluster323	1
cluster355	1
cluster419	1
cluster451	1
cluster483	1
cluster515	1
cluster547	1
cluster675	1
cluster707	1
cluster739	1
cluster771	1
cluster803	1
cluster835	1
cluster3	1
cluster131	1
cluster252	1
cluster284	1
cluster316	1
cluster380	1
cluster412	1
cluster444	1
cluster476	1
cluster508	1
cluster540	1
cluster572	1
cluster636	1
cluster668	1
cluster700	1
cluster732	1
cluster28	1
cluster60	1
cluster124	1
cluster149	1
cluster181	1

cluster277	1
cluster373	1
cluster469	1
cluster501	1
cluster533	1
cluster565	1
cluster693	1
cluster725	1
cluster757	1
cluster853	1
cluster53	1
cluster142	1
cluster174	1
cluster334	1
cluster430	1
cluster462	1
cluster494	1
cluster526	1
cluster558	1
cluster590	1
cluster814	1
cluster846	1
cluster14	1
cluster46	1
cluster110	1
cluster135	1
cluster167	1
cluster199	1
cluster263	1
cluster295	1
cluster327	1
cluster359	1
cluster487	1
cluster551	1
cluster583	1
cluster679	1
cluster711	1
cluster743	1
cluster807	1
cluster39	1
cluster71	1
cluster160	1
cluster224	1
cluster256	1

cluster320	1
cluster384	1
cluster416	1
cluster480	1
cluster512	1
cluster544	1
cluster608	1
cluster640	1
cluster672	1
cluster704	1
cluster64	1
cluster128	1
cluster185	1
cluster217	1
cluster345	1
cluster537	1
cluster569	1
cluster633	1
cluster697	1
cluster761	1
cluster793	1
cluster857	1
cluster89	1
cluster121	1
cluster178	1
cluster210	1
cluster306	1
cluster370	1
cluster434	1
cluster466	1
cluster498	1
cluster530	1
cluster562	1
cluster626	1
cluster658	1
cluster690	1
cluster722	1
cluster754	1
cluster850	1
cluster18	1
cluster50	1
cluster82	1
cluster114	1
cluster139	1

cluster171	1
cluster203	1
cluster235	1
cluster331	1
cluster427	1
cluster491	1
cluster555	1
cluster587	1
cluster619	1
cluster683	1
cluster715	1
cluster779	1
cluster811	1
cluster843	1
cluster11	1
cluster75	1
cluster164	1
cluster228	1
cluster356	1
cluster388	1
cluster420	1
cluster452	1
cluster484	1
cluster516	1
cluster580	1
cluster676	1
cluster740	1
cluster772	1
cluster804	1
cluster836	1
cluster4	1
cluster36	1
cluster68	1
cluster100	1
cluster132	1

Tabla 21 Clusters Pehash-Imphash-Icon_Label

Para comprender de mejor forma los resultados del algoritmo luego de añadir el atributo ICON_LABEL obtendremos las métricas de Precision y Recall de los dos nuevos clustering el Pehash-Imphash y el Pehash-Imphash-Icon_label, estos datos se presentan en la siguiente Tabla 22.

Característica	PRECISSION	RECALL	F-MEASURE
Pehash-Imphash	99,76%	8,73%	16,06
Pehash-Imphash-Icon_label	99,85%	43,44%	60,54

Tabla 22 Métricas de los nuevos clústers

La Tabla anterior nos indica que la precisión se mantiene alta y el Recall bajo como en los casos anteriores; al aumentar el atributo ICON_LABEL el Precisión aumenta en 0.09% y el Recall aumenta en un 34,71%.

Para agrupar los clústers que en la tabla anterior el Recall nos indica que están separados, vamos ya a clasificarlos con las etiquetas que tienen en el atributo FAMILY de la actual clasificación de MALICIA. Para esto implementaremos un algoritmo que les asigne el nombre de la familia de acuerdo a la mayor cantidad de elementos de una misma familia. Es decir, si un clúster tiene 10 elementos de los cuales 8 pertenecen a la familia A y los otros 2 pertenecen a la familia B, a todo el clúster se le asignará la etiqueta de familia A. El script que implementa este algoritmo lo hemos llamado "MajorityVoting.py" el cual se presenta en la siguiente tabla:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"" Author: Richard Rivera
    Script for parameters for malware Clustering
""
import MySQLdb as mdb
import sys
from collections import Counter
import os
import hashlib
import pefile
import bitstring
import string
import bz2

db = mdb.connect('localhost', 'user', 'password', 'RELEASE1.0')
cursorC = db.cursor()
cursorUpdate= db.cursor()
cursorJ=db.cursor()
clusterID=1
#getting n from the valid samples.
cursorC.execute("SELECT DISTINCT CLUSTER2 FROM FILE_TMPC")
for rowC in cursorC.fetchall():
    cursorJ.execute("SELECT FAMILY FROM FILE_TMPC WHERE CLUSTER2
='{}'".format(rowC[0]))
    clusters=list(cursorJ.fetchall())
    data = Counter(clusters)
    (familyNew,)= data.most_common(1)[0][0]
    if familyNew==None:
        familyNew="clusterTFMRR{}".format(clusterID)
        cursorUpdate.execute("UPDATE FILE_TMPC SET FAMILY2 ='{}'
WHERE CLUSTER2='{}'".format(familyNew,rowC[0]))
        db.commit()
        clusterID=clusterID+1
    else:
        cursorUpdate.execute("UPDATE FILE_TMPC SET FAMILY2 ='{}'
WHERE CLUSTER2='{}'".format(familyNew,rowC[0]))
```

```
        db.commit()
db.close()
```

Tabla 23 Script MajorityVoting.py

El algoritmo Majority voting o Weighted Majority (Lam & Suen, 1997) para encontrar la característica con mayor número de ocurrencias dentro de un clúster puede ser implementado de varias formas. En este trabajo se optó por utilizar el método "`most_common`" de la librería Collections de Python que ya implementa este algoritmo (Python Software Foundation, 2014).

El resultado de este nuevo script para poner una etiqueta a las nuevas familias tomando en cuenta las etiquetas actuales de MALICIA, es un total de 418 familias, reduciendo el número de clústers a la mitad del último clustering de Pehash_Imphash_Icon_Label.

Cluster_ID	#
winwebsec	5875
zbot	2207
zeroaccess	1328
securityshield	151
smarthdd	71
cridex	62
harebot	52
clusterTFMRR54	49
clusterTFMRR74	47
CLUSTER:85.93.17.123	45
clusterTFMRR324	38
clusterTFMRR309	33
clusterTFMRR318	33
clusterTFMRR223	33
clusterTFMRR281	30
CLUSTER:newavr	29
clusterTFMRR228	27
clusterTFMRR289	26
clusterTFMRR314	26
CLUSTER:chapterleomemorykombo.eu	25
clusterTFMRR6	24
clusterTFMRR301	23
clusterTFMRR302	23
clusterTFMRR298	22
clusterTFMRR279	22
clusterTFMRR270	21
clusterTFMRR45	21
clusterTFMRR328	20
CLUSTER:46.105.131.121	20
clusterTFMRR99	19
clusterTFMRR70	19
clusterTFMRR337	18
clusterTFMRR248	18
clusterTFMRR340	18
clusterTFMRR267	17
clusterTFMRR307	16
clusterTFMRR294	15
clusterTFMRR235	15
clusterTFMRR58	15
clusterTFMRR333	14
CLUSTER:positivtkn.in.ua	14
clusterTFMRR103	14
clusterTFMRR39	13
clusterTFMRR23	13
clusterTFMRR356	13
clusterTFMRR354	12
clusterTFMRR327	11
clusterTFMRR312	11
clusterTFMRR28	11
reveton	10
clusterTFMRR238	10
CLUSTER:gdata	10
clusterTFMRR101	10
spyeye-ep	9
clusterTFMRR233	8
clusterTFMRR361	8
clusterTFMRR346	8
clusterTFMRR41	8
clusterTFMRR329	7
clusterTFMRR287	7
clusterTFMRR24	7
CLUSTER:foreign	7
clusterTFMRR265	6
clusterTFMRR242	6
clusterTFMRR331	6
clusterTFMRR276	6
clusterTFMRR344	6
clusterTFMRR249	5
clusterTFMRR40	5
clusterTFMRR178	5
clusterTFMRR48	5
ransomNoaouy	5
clusterTFMRR288	5

clusterTFMRR320	5
clusterTFMRR336	5
ramnit	5
clusterTFMRR86	5
clusterTFMRR173	4
clusterTFMRR253	4
clusterTFMRR293	4
clusterTFMRR317	4
clusterTFMRR52	4
clusterTFMRR150	4
clusterTFMRR198	4
clusterTFMRR195	4
clusterTFMRR259	4
clusterTFMRR263	4
clusterTFMRR295	4
CLUSTER:91.234.32.10	4
clusterTFMRR232	4
clusterTFMRR352	4
clusterTFMRR360	4
CLUSTER:contacts	4
CLUSTER:online-police.com	3
clusterTFMRR157	3
clusterTFMRR241	3
clusterTFMRR341	3
clusterTFMRR124	3
clusterTFMRR190	3
clusterTFMRR262	3
clusterTFMRR310	3
fakeav-webprotection	3
fakeav-rena	3
clusterTFMRR100	3
clusterTFMRR243	3
clusterTFMRR247	3
clusterTFMRR275	3
clusterTFMRR335	3
clusterTFMRR339	3
CLUSTER:autoit-plus	3
clusterTFMRR160	3
clusterTFMRR164	3
clusterTFMRR172	3
clusterTFMRR260	3
clusterTFMRR264	3
clusterTFMRR20	3
clusterTFMRR165	2
clusterTFMRR193	2
clusterTFMRR257	2
clusterTFMRR277	2
clusterTFMRR325	2
clusterTFMRR38	2
clusterTFMRR50	2
clusterTFMRR67	2
clusterTFMRR71	2

clusterTFMRR83	2
clusterTFMRR95	2
clusterTFMRR104	2
clusterTFMRR142	2
clusterTFMRR162	2
clusterTFMRR174	2
clusterTFMRR230	2
clusterTFMRR246	2
clusterTFMRR250	2
clusterTFMRR282	2
clusterTFMRR330	2
cutwail	2
clusterTFMRR30	2
clusterTFMRR51	2
clusterTFMRR53	2
clusterTFMRR64	2
clusterTFMRR72	2
CLUSTER:floppies	2
clusterTFMRR92	2
clusterTFMRR117	2
clusterTFMRR155	2
clusterTFMRR167	2
clusterTFMRR175	2
clusterTFMRR183	2
clusterTFMRR207	2
clusterTFMRR211	2
clusterTFMRR299	2
clusterTFMRR303	2
clusterTFMRR351	2
clusterTFMRR359	2
clusterTFMRR367	2
clusterTFMRR27	2
clusterTFMRR46	2
clusterTFMRR65	2
clusterTFMRR78	2
clusterTFMRR85	2
clusterTFMRR106	2
clusterTFMRR134	2
clusterTFMRR176	2
clusterTFMRR184	2
clusterTFMRR216	2
clusterTFMRR268	2
clusterTFMRR280	2
clusterTFMRR296	2
clusterTFMRR308	2
clusterTFMRR115	2
clusterTFMRR128	1
clusterTFMRR131	1
clusterTFMRR138	1
clusterTFMRR145	1
clusterTFMRR149	1
clusterTFMRR153	1

clusterTFMRR161	1
clusterTFMRR169	1
clusterTFMRR177	1
clusterTFMRR181	1
clusterTFMRR185	1
clusterTFMRR189	1
clusterTFMRR197	1
clusterTFMRR201	1
clusterTFMRR205	1
clusterTFMRR209	1
clusterTFMRR213	1
clusterTFMRR217	1
clusterTFMRR221	1
clusterTFMRR225	1
clusterTFMRR229	1
clusterTFMRR237	1
clusterTFMRR245	1
clusterTFMRR261	1
clusterTFMRR269	1
clusterTFMRR273	1
clusterTFMRR285	1
clusterTFMRR297	1
clusterTFMRR305	1
clusterTFMRR313	1
clusterTFMRR321	1
clusterTFMRR345	1
clusterTFMRR349	1
clusterTFMRR353	1
clusterTFMRR357	1
clusterTFMRR365	1
clusterTFMRR369	1
clusterTFMRR1	1
clusterTFMRR7	1
clusterTFMRR10	1
clusterTFMRR12	1
clusterTFMRR14	1
clusterTFMRR18	1
clusterTFMRR21	1
dprn	1
clusterTFMRR26	1
clusterTFMRR29	1
CLUSTER:blackandwhite	1
CLUSTER:azonpowzanadinoar.com	1
CLUSTER:184.82.148.49	1
clusterTFMRR44	1
clusterTFMRR47	1
clusterTFMRR56	1
clusterTFMRR60	1
clusterTFMRR63	1
clusterTFMRR73	1
clusterTFMRR77	1
clusterTFMRR80	1

clusterTFMRR87	1
clusterTFMRR91	1
ufasoft-bitcoin	1
clusterTFMRR108	1
clusterTFMRR112	1
clusterTFMRR116	1
clusterTFMRR120	1
clusterTFMRR132	1
clusterTFMRR135	1
clusterTFMRR139	1
clusterTFMRR146	1
clusterTFMRR154	1
clusterTFMRR158	1
clusterTFMRR166	1
clusterTFMRR170	1
clusterTFMRR182	1
clusterTFMRR186	1
clusterTFMRR194	1
clusterTFMRR202	1
clusterTFMRR206	1
clusterTFMRR210	1
clusterTFMRR214	1
clusterTFMRR218	1
clusterTFMRR222	1
clusterTFMRR226	1
clusterTFMRR234	1
clusterTFMRR254	1
clusterTFMRR258	1
clusterTFMRR266	1
clusterTFMRR274	1
clusterTFMRR278	1
clusterTFMRR286	1
clusterTFMRR290	1
clusterTFMRR306	1
clusterTFMRR322	1
clusterTFMRR326	1
clusterTFMRR334	1
clusterTFMRR338	1
clusterTFMRR342	1
clusterTFMRR350	1
clusterTFMRR358	1
clusterTFMRR362	1
clusterTFMRR366	1
clusterTFMRR370	1
CLUSTER:mindamura.com	1
CLUSTER:217.20.115.99	1
CLUSTER:eikons.com	1
clusterTFMRR15	1
clusterTFMRR19	1
clusterTFMRR22	1
clusterTFMRR25	1
CLUSTER:fadedtext	1

CLUSTER:halfmoon	1
clusterTFMRR34	1
clusterTFMRR57	1
clusterTFMRR61	1
clusterTFMRR68	1
clusterTFMRR81	1
clusterTFMRR84	1
clusterTFMRR88	1
clusterTFMRR96	1
clusterTFMRR102	1
clusterTFMRR105	1
clusterTFMRR109	1
clusterTFMRR113	1
clusterTFMRR121	1
clusterTFMRR125	1
clusterTFMRR129	1
clusterTFMRR133	1
clusterTFMRR136	1
clusterTFMRR140	1
clusterTFMRR143	1
clusterTFMRR147	1
clusterTFMRR151	1
clusterTFMRR159	1
clusterTFMRR163	1
clusterTFMRR171	1
clusterTFMRR179	1
clusterTFMRR187	1
clusterTFMRR191	1
clusterTFMRR199	1
clusterTFMRR203	1
clusterTFMRR215	1
clusterTFMRR219	1
clusterTFMRR227	1
clusterTFMRR231	1
clusterTFMRR239	1
clusterTFMRR251	1
clusterTFMRR255	1
clusterTFMRR271	1
clusterTFMRR283	1
clusterTFMRR291	1
clusterTFMRR311	1
clusterTFMRR315	1
clusterTFMRR319	1
clusterTFMRR323	1
clusterTFMRR343	1
clusterTFMRR347	1
clusterTFMRR355	1
clusterTFMRR363	1
clusterTFMRR2	1
clusterTFMRR4	1
russkill	1
clusterTFMRR11	1

clusterTFMRR13	1
clusterTFMRR16	1
CLUSTER:colorballs	1
clusterTFMRR32	1
clusterTFMRR35	1
clusterTFMRR36	1
clusterTFMRR42	1
CLUSTER:up2x.com	1
CLUSTER:in7cy	1
clusterTFMRR62	1
clusterTFMRR69	1
CLUSTER:justontime-12.com	1
clusterTFMRR75	1
clusterTFMRR82	1
clusterTFMRR89	1
clusterTFMRR93	1
clusterTFMRR97	1
clusterTFMRR110	1
clusterTFMRR114	1
clusterTFMRR118	1
clusterTFMRR122	1
clusterTFMRR126	1
clusterTFMRR130	1
clusterTFMRR137	1
clusterTFMRR141	1
clusterTFMRR144	1
clusterTFMRR148	1
clusterTFMRR152	1
clusterTFMRR156	1
clusterTFMRR168	1
clusterTFMRR180	1
clusterTFMRR188	1
clusterTFMRR192	1
clusterTFMRR196	1
clusterTFMRR200	1
clusterTFMRR204	1
clusterTFMRR208	1
clusterTFMRR212	1
clusterTFMRR220	1
clusterTFMRR224	1
clusterTFMRR236	1
clusterTFMRR240	1
clusterTFMRR244	1
clusterTFMRR252	1
clusterTFMRR256	1
clusterTFMRR272	1
clusterTFMRR284	1
clusterTFMRR292	1
clusterTFMRR300	1
clusterTFMRR304	1
clusterTFMRR316	1
clusterTFMRR332	1

clusterTFMRR348	1
clusterTFMRR364	1
clusterTFMRR368	1
clusterTFMRR3	1
clusterTFMRR5	1
clusterTFMRR8	1
clusterTFMRR9	1
clusterTFMRR17	1
CLUSTER:m9swachu.be	1
clusterTFMRR31	1
clusterTFMRR33	1
CLUSTER:bundlemonkey.com	1
clusterTFMRR37	1
CLUSTER:whatismyip.com	1
clusterTFMRR43	1
clusterTFMRR49	1
CLUSTER:paydayloatnsstc.ru	1

clusterTFMRR55	1
clusterTFMRR59	1
CLUSTER:clarkclark	1
clusterTFMRR66	1
CLUSTER:mycomputer	1
clusterTFMRR76	1
clusterTFMRR79	1
CLUSTER:mergenew	1
clusterTFMRR90	1
clusterTFMRR94	1
clusterTFMRR98	1
clusterTFMRR107	1
clusterTFMRR111	1
clusterTFMRR119	1
clusterTFMRR123	1
clusterTFMRR127	1

Tabla 24 Clustering Majority Voting

De los resultados de la tabla anterior ya se puede notar como se han agrupado las familias más representativas teniendo cantidades iguales o similares a las actuales de MALICIA. Para representar mejor estos resultados de este nuevo clustering llamado Majority_Voting también obtendremos los valores de Precision y Recall los cuales se presentan en la siguiente tabla:

Característica	PRECISION	RECALL	F- MEASURE
Majority_Voting	99,15%	99,32%	99,23

Tabla 25 Métricas Clustering Majority Voting

Como se muestra en la tabla anterior al utilizar el último algoritmo de Majority Voting el Recall que en todos los casos se encontraba a menos del 50% ahora presenta un 99,32%. En este nuevo clustering tenemos 418 familias de las cuales al menos 171 tienen más de un fichero ejecutable.

Como se comentó en el inicio de este capítulo los dos objetivos principales de este trabajo eran encontrar características estáticas de los ficheros ejecutables que nos permitan clasificar los malware de MALICIA, estas características fueron el Imphash, el Pehash y el ICON_LABEL esta última tomada de las características que ya tenía MALICIA, y el segundo objetivo era mejorar el índice de clasificación de MALICIA, que antes tenía un 87,24% y ahora se ha puesto etiquetas de familias a todos los elementos con lo que se podría decir que está clasificada en un 100%, pero tomado en cuenta que se crearon 227 nuevas familias con un solo fichero ejecutable la clasificación real se establecería en un 98% mejorando en un 10,76%.

5Análisis de Resultados

En este apartado se presenta de forma consolidada los resultados obtenidos en éste trabajo de investigación. El objetivo de este trabajo era analizar características estáticas que se pueden obtener de los ficheros ejecutables para poder realizar una clasificación de los malware de MALICIA, tratando de aumentar el índice de clasificación actual.

5.1 Comparativa Pehash – Imphash

Las características estáticas de ficheros ejecutables seleccionadas para el análisis de este trabajo fueron el Imphash y el Pehash. Se obtuvo estas características de los ficheros de MALICIA y se generó un clustering para cada una de éstas. Posteriormente, se validó estos clústers contra el clustering de referencia que era la actual clasificación de MALICIA. Estos resultados se presentan en la siguiente tabla. En esta comparativa se añade la métrica F-Measure mencionada en el apartado 4.3 de este trabajo.

Característica	Precision	Recall	F-Measuere
Imphash	**99,76**	**5,69**	**10,77%**
Pehash	**99,99**	**8,05**	**14,90%**

Tabla 26 Comparativa Pehash-Imphash

Estos resultados nos indican que la mejor característica de las dos es el Pehash por tener un mayor valor en la métrica F-Measure, y al tener un Clustering de referencia esta característica es de gran utilidad para encontrar posibles errores en la clasificación del Clustering de referencia como sucedió en el desarrollo de este trabajo.

La característica Pehash solo presento inconvenientes sobre 4 de los ficheros binarios analizados de los 11.363 ficheros analizados, por otra parte Imphash presento inconvenientes en estos 4 ficheros y en 133 más que tenían la tabla de importación vacía y puesto que el Imphash utiliza esta sección del PE para sacar el hash estos ficheros no generaron ningún Imphash, esto es algo a tener en cuenta ahora que las personas que desarrollan malware pueden valerse de distintas técnicas para ocultar la tabla de importación hasta que sea necesaria en tiempo de ejecución.

5.2 Clasificación

Para poder realizar la clasificación de MALICIA con las nuevas características estáticas y tomando La característica ICON_LABEL de la actual clasificación de MALICIA se realizaron varios procesos empezando por la clasificación individual de Imphash y Pehash. Posteriormente, se juntó estas en el clustering Pehash-Imphash con el algoritmo clustering agresivo, después se añadió la característica ICON_LABEL y se generó el Clustering Pehash-Imphash-Icon_label, el cual aún tenía un Recall por debajo del 50%. Finalmente, para aumentar el Recall se utilizó el algoritmo de Majority Voting generando el clustering final con el mismo nombre Majority Voting. Cabe aclarar que este último clustering se generó a partir del clustering Pehash-Imphash-Icon_label. Los resultados con la métrica F-Measure se presentan a continuación:

Característica	#clústers	#singleton clústers	Precision	Recall	F-Measure
Imphash	2028	912	99,76	5,69	10,77%
Pehash	1500	804	99,99	8,05	14,90%
Pehash-Imphash	1136	570	99,76%	8,73%	16,06%
Pehash-Imphash-Icon_label	858	495	99,85%	43,44%	60,54%
Pehash-Imphash-Icon_label (Majority Voting)	418	247	99,15%	99,32%	99,23%

Tabla 27 Resultado final de clasificaciones

Todas estas clasificaciones fueron realizadas sobre un total 11358 ficheros ejecutables PE. La tabla anterior presenta los resultados donde se puede apreciar cómo va mejorando el F-Measure a lo largo de todo este proceso hasta alcanzar un notable **99,32%**. En el último paso hay un leve decremento del Precision, esto es debido a que varias familias del clustering de referencia en el clustering de resultado se unifican y también a los pequeños errores o excepciones que encontramos en el clustering de referencia. Adicionalmente, se puede notar como en cada paso del proceso el número de clústers va decreciendo hasta llegar a 418 clústers, de igual forma para los Singleton clústers, que son los clústers con un solo elemento. Al final tenemos únicamente 247 ficheros PE en esta categoría.

6Conclusiones y Trabajo futuro

6.1 Conclusiones

El principal objetivo de este trabajo era analizar características estáticas que se puede obtener de los ficheros ejecutables para poder realizar una clasificación de los malware de MALICIA, tratando de aumentar el índice de clasificación actual de MALICIA.

- Las características estáticas analizadas fueron Imphash, Pehash e ICON_LABEL. Esta última tomada de las características que ya tenía MALICIA. Con estas características estáticas de los PE se trataba de mejorar el índice de clasificación de MALICIA que antes tenía un 87,24%. Ahora, se ha puesto etiquetas de familias a todos los elementos con lo que se podría decir que está clasificada en un 100%, pero tomado en cuenta que se crearon 227 nuevas familias con un sólo fichero ejecutable, la clasificación real se establecería en un 98% mejorando la clasificación en un 10,76%.
- La métrica F-Measure que agrupa el Precision y el Recall nos sirve para saber cuál técnica de clasificación tiene más validez. Al obtener un F-Measure para el clustering con Majority Voting con un valor de 99,23% se puede concluir que todo el proceso y los clustering anteriores que sirvieron para generar este, se condujeron por el camino adecuado. Por lo tanto se puede utilizar este proceso para futuras clasificaciones donde se cuente con un Clustering de Referencia.
- Al no tener un clustering de referencia se puede utilizar como punto de partida para la clasificación el clustering generado por la característica estática del Pehash o el Pehash-Imphash más el ICON_LABEL que en cuanto a clasificación es más eficiente de acuerdo a los resultados de este trabajo.
    - Juan Caballero en su calidad de director de este trabajo y representando al proyecto MALICIA Project (Imdea Software Institute, 2013) contaba con otro conjunto de datos de malware denominado “ssl_malware” que no estaban clasificados, es decir no se tenía un clustering de referencia. De forma adicional a este trabajo se solicito obtener las características Imphash y Pehash de estos *1.203* ficheros binarios PE y adicional al

archivo generado con el Sha1, Md5, Pehash e Imphash se realizó el análisis de la siguiente tabla.

Característica	#clústers	Single clúster
Imphash	163	125
Pehash	102	74
Pehash-Imphash	102	74

No se nota una mejora en el clustering Pehash-Imphash sobre el clustering de Pehash pero es un buen punto de partida para clasificar las primeras familias de estos ficheros ejecutables utilizando únicamente el clustering de Pehash.

- Las métricas RS, JC y FM además de proporcionar un índice de valides del clustering nos permiten detectar fallos en el clustering de referencia que se pueden corregir para obtener mejores métricas de Imphash y Pehash.
- El lenguaje Python presenta varias facilidades para realizar este tipo de investigaciones, siendo muy intuitivo de aprender. Además cuenta con muchas librerías de código libre realizadas por investigadores o entusiastas del lenguaje que facilitan el trabajo.
- El malware ha incrementado su complejidad, variedad y las técnicas más sofisticadas están ganando un amplio uso, aumentando el esfuerzo necesario para combatir este problema. Los productores de malware seguirán utilizando todos los medios necesarios a fin de conseguir los objetivos deseados. Esto implica buscar nuevos caminos sin dejar de utilizar cualquier otro método disponible, generando así más retos a los investigadores interesados en el análisis y clasificación del Malware.

6.2 Trabajo Futuro

Esta sección se presenta los trabajos futuros que se puede seguir investigando o implementando a partir de este trabajo.

- Todos los procesos presentados hasta llegar al último clustering presentado se podrían implementar en un único proceso, eliminando los procesos repetidos. Estas fueron lecciones aprendidas en este trabajo que se podrían optimizar, realizando pruebas para muestras pequeñas antes de realizar sobre todos los ficheros si la muestra es grande.

- El proyecto MALICIA (Imdea Software Institute, 2013) ha publicado varios artículos sobre la clasificación de malware, en especial el artículo de la herramienta FIRMA (Rafique & Caballero, 2013). Se podría estudiar la posibilidad de añadir a esta herramienta el análisis de las características estáticas de forma automática para mejorar las futuras clasificaciones de malware.

- A demás de las características estáticas estudiadas sobre los ficheros PE, profundizar en este tema podría permitir encontrar otras características estáticas que presenten altos índices de Precision y Recall.

7Referencias

Aditya, G. (2010). Exhaustive Statistical Analysis for Detection of Metamorphic Malware. *Master's Projects*, 66. Obtenido de http://scholarworks.sjsu.edu/etd_projects/66

Aldeid. (2013). *PEiD*. Obtenido de Aldeid: http://www.aldeid.com/wiki/PEiD

Alvarez, V. M. (s.f.). *Yara The pattern matching swiss knife for malware researchers (and everyone else)*. Obtenido de http://plusvic.github.io/yara/

Apache Web server. (s.f.). Obtenido de http://httpd.apache.org.

Asaf Shabtai, R. M. (2009). Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. *Information Security Technical Report, 14*, 16-29.

Canonical. (2014). *Ubuntu The leading OS for PC, tablet, phone and cloud*. Obtenido de http://www.ubuntu.com/

Carrera, E. (26 de 12 de 2005). *Portable Executable File Format.* Recuperado el 22 de 04 de 2014, de http://www.openrce.org/reference_library/files/reference/PE%20Format.pdf

Carrera, E. (2013). *pefile is a Python module to read and work with PE (Portable Executable) files*. Obtenido de Pefile: https://code.google.com/p/pefile/

Clark, C. (2014). *YaraGenerator - Automatic CND Signature Generatio*. Obtenido de YaraGenerator: https://yaragenerator.com/about/

Cohen, F. (1985). *Computer viruses. PhD thesis.* University of Southern California.

dechile.net. (2010). *Etimología del malware*. Recuperado el Mayo de 2014, de http://etimologias.dechile.net/?malware

EC-Council. (2013). *Ethical Hacking and Countermeasuresv8.*

Higuera, J. B. (2013). Análisis de malware. *Apuntes del Profesor-Asignatura Seguridad en el Sofware. Universidad Internacional de la Rioja*. Rioja, España.

Imdea Software Institute. (2013). *Malicia Project*. Obtenido de http://malicia-project.com/

K. Rieck, T. H. (2008). Learning and Classification of Malware Behavior. *DIMVA*, 108-125.

kaspersky. (2010). *¿Qué es un troyano?* Obtenido de http://www.kaspersky.es/internet-security-center/threats/trojans

Lam, L., & Suen, C. Y. (1997). Application of Majority Voting to Pattern Recognition: An Analysis of Its Behavior and Performance. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS*.

M. Halkidi, Y. B. (2001). On clustering validation techniques. *J. Intell. Inf. Syst.*, 107–145.

Mandiant Corporation. (2014). *Tracking Malware with Import Hashing.* Obtenido de Mandiant: https://www.mandiant.com/blog/tracking-malware-import-hashing/

McAfee. (Abril de 2006). *Rootkits, Part 1 of 3: The Growing Threat.* Obtenido de http://web.archive.org/web/20060823090948/http://www.mcafee.com/us/local_content/white_papers/threat_center/wp_akapoor_rootkits1_en.pdf

Microsoft. . (s.f.). *MSDN: TheMicrosoft developer network.* Obtenido de http://msdn.microsoft.com.

Microsoft1. (3 de Febrero de 2013). *Microsoft Portable Executable and Common Object File Format Specification.* Recuperado el 29 de Marzo de 2014, de http://msdn.microsoft.com/en-us/library/gg463119.aspx

Microsoft2. (2014). *Metadatos y la estructura del archivo PE*. Recuperado el 01 de 05 de 2014, de Microsoft Developer Network MSDN: http://msdn.microsoft.com/es-es/library/8dkk3ek4.aspx

Moessing, L. (14 de Agosto de 2007). *PE File Format in XML.* Recuperado el 22 de 04 de 2014, de http://www.openrce.org/reference_library/files/reference/Portable_Executable_Format.xml

Nappa, A., Rafique, M. Z., & Caballero, J. (2013). Driving in the Cloud: An Analysis of Drive-by Download Operations and Abuse Reporting. *Proceedings of the 10th Conference on Detection of Intrusions and Malware & Vulnerability Assessment Berlin DE*.

Oracle, C. (2014). *MySQL The world's most popular open source database*. Obtenido de http://www.mysql.com/

Panda Security. (Mayo de 2010). *Troyano*. Obtenido de http://www.pandasecurity.com/spain/homeusers/security-info/classic-malware/trojan/

Perdisci, R., & ManChon, U. (2008). VAMO: Towards a Fully Automated Malware Clustering Validity Analysis. *ACSAC*.

Pietrek, M. (1994). Peering Inside the PE: A Tour of the Win32 Portable Executable File Format. *Microsoft Systems Journal, 9*(3), 15-34.

Python Software Foundation. (2014). *Python 3.4.1 documentation*. Obtenido de https://docs.python.org/3/

Rafique, Z., & Caballero, J. (2013). FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors.

Rutaa, D., & Gabry, B. (2005). Classifier selection for majority voting. *Information Fusion*, 63–81.

Saeed, M., Lindskog, D., Zavarsky, P., & Ruhl, R. (2013). Two techniques for detecting packed portable executable files. *Information Society (i-Society), 2013 International Conference on*, 22-26.

Snider, D. (7 de Mayo de 2013). *Data Mining Clustering Example in SQL Server Analysis Services SSAS*. Obtenido de http://www.mssqltips.com/sqlservertip/2993/data-mining-clustering-example-in-sql-server-analysis-services-ssas/

Totalhash. (s.f.). *#totalhash Malware Analysis Database*. Obtenido de http://totalhash.com/

U. Bayer, P. M. (2009). Scalable, behavior-based malware clustering. *NDSS*.

Virustotal. (s.f.). *Virustotal*. Obtenido de https://www.virustotal.com/

Wang, T.-Y., Wu, C.-H., & Hsieh, C.-C. (2009). Detecting unknown malicious executables using portable executable headers. *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on, doi:10.1109/NCM.2009.385*, 278-284.

Wang, T.-Y., Wu, C.-H., & Hsieh, C.-C. (2009). Detecting Unknown Malicious Executables Using Portable Executable Headers. *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*.

Wei, W. (s.f.). *PE Trojan Detection Based on the Assessment of Static File Features.* Obtenido de Antiy Labs: http://www.antiy.net/media/slides/trojan-detection_slide.pdf

Wicherski, G. (2013). *peHash: A Novel Approach to Fast Malware Clustering.* Obtenido de www.usenix.org: https://www.usenix.org/legacy/events/leet09/tech/full_papers/wicherski/wicherski.pdf

Wikipedia. (s.f.). Obtenido de http://www.wikipedia.org.Wine. (s.f.). Obtenido de http://www.winehq.org/.

Anexos

Anexo 1 Script completo para generar el Imphash y el Pehash de un PE

```
#!/usr/bin/python
“” Author: Richard Rivera
    Script for obtain the md5, sha1, Imphash, Pehash for a specific
directory
    Execution example: $ python Hashes.py
    Hashes.py don’t need parameters

from __future__ import division
import os
import sys
import hashlib
import pefile
import bitstring
import string
import bz2

def recursive():
    with open(“output”, “w”) as a:
        for path, subdirs, files in os.walk (‘/home/binarios/’):
            for filename in files:
                #md5
                with open(os.path.join(path,filename),’r’) as f:
                    for chunk in iter(lambda: f.read(), b”):
                        md5 = hashlib.md5()
                        md5.update(chunk)
                md5Hash=md5.hexdigest()
                #sha1
                with open(os.path.join(path,filename),’r’) as f:
                    for chunk in iter(lambda: f.read(), b”):
                        sha1= hashlib.sha1()
                        sha1.update(chunk)
                sha1Hash=sha1.hexdigest()
                #imphash
                try:
                    p = pefile.PE(os.path.join(path,filename))
                    imphash = p.get_imphash()
                    #control for PE file with empty import table
                    if imphash==””:
                        imphash=”null”
                except:
                    #control for not PE file
                    pass
                    imphash=”null”
                #pehash to file
                try:
pehash=pehash_To_File(os.path.join(path,filename))

                except:
                    pass
                    pehash=”null”
                #print to file
<<filename>>,<<md5>>,<<sha1>>,<<imphash>>,<<pehash>>
```

```
                f =
("{},{},{},{},{}".format(os.path.join(filename),md5Hash,sha1Hash,imp
hash,pehash))
                    a.write(str(f) + os.linesep)
#pehash
""
Script published by Totalhash.com as pehash.py
adapted as a function for Hashes.py by Richard Rivera
""
def pehash_To_File(filename):
    exe = pefile.PE(filename)
    #image characteristics
    img_chars =
bitstring.BitArray(hex(exe.FILE_HEADER.Characteristics))
    #pad to 16 bits
    img_chars = bitstring.BitArray(bytes=img_chars.tobytes())
    img_chars_xor = img_chars[0:7] ^ img_chars[8:15]
    #start to build pehash
    pehash_bin = bitstring.BitArray(img_chars_xor)

    #subsystem -
    sub_chars = bitstring.BitArray(hex(exe.FILE_HEADER.Machine))
    #pad to 16 bits
    sub_chars = bitstring.BitArray(bytes=sub_chars.tobytes())
    sub_chars_xor = sub_chars[0:7] ^ sub_chars[8:15]
    pehash_bin.append(sub_chars_xor)

    #Stack Commit Size
    stk_size =
bitstring.BitArray(hex(exe.OPTIONAL_HEADER.SizeOfStackCommit))
    stk_size_bits = string.zfill(stk_size.bin, 32)
    #now xor the bits
    stk_size = bitstring.BitArray(bin=stk_size_bits)
    stk_size_xor = stk_size[8:15] ^ stk_size[16:23] ^
stk_size[24:31]
    #pad to 8 bits
    stk_size_xor = bitstring.BitArray(bytes=stk_size_xor.tobytes())
    pehash_bin.append(stk_size_xor)

    #Heap Commit Size
    hp_size =
bitstring.BitArray(hex(exe.OPTIONAL_HEADER.SizeOfHeapCommit))
    hp_size_bits = string.zfill(hp_size.bin, 32)
    #now xor the bits
    hp_size = bitstring.BitArray(bin=hp_size_bits)
    hp_size_xor = hp_size[8:15] ^ hp_size[16:23] ^ hp_size[24:31]
    #pad to 8 bits
    hp_size_xor = bitstring.BitArray(bytes=hp_size_xor.tobytes())
    pehash_bin.append(hp_size_xor)

    #Section chars
    for section in exe.sections:
        #virutal address
        sect_va =  bitstring.BitArray(hex(section.VirtualAddress))
        sect_va = bitstring.BitArray(bytes=sect_va.tobytes())
        pehash_bin.append(sect_va)

        #rawsize
        sect_rs =  bitstring.BitArray(hex(section.SizeOfRawData))
        sect_rs = bitstring.BitArray(bytes=sect_rs.tobytes())
        sect_rs_bits = string.zfill(sect_rs.bin, 32)
```

```
        sect_rs = bitstring.BitArray(bin=sect_rs_bits)
        sect_rs = bitstring.BitArray(bytes=sect_rs.tobytes())
        sect_rs_bits = sect_rs[8:31]
        pehash_bin.append(sect_rs_bits)

        #section chars
        sect_chars =
bitstring.BitArray(hex(section.Characteristics))
        sect_chars = bitstring.BitArray(bytes=sect_chars.tobytes())
        sect_chars_xor = sect_chars[16:23] ^ sect_chars[24:31]
        pehash_bin.append(sect_chars_xor)

        #entropy calulation
        address = section.VirtualAddress
        size = section.SizeOfRawData
        raw = exe.write()[address+size:]
        if size == 0:
            kolmog = bitstring.BitArray(float=1, length=32)
            pehash_bin.append(kolmog[0:7])
            continue
        bz2_raw = bz2.compress(raw)
        bz2_size = len(bz2_raw)
        #k = round(bz2_size / size, 5)
        k = bz2_size / size
        kolmog = bitstring.BitArray(float=k, length=32)
        pehash_bin.append(kolmog[0:7])

    m = hashlib.sha1()
    m.update(pehash_bin.tobytes())
    return m.hexdigest()

def main(args):

    recursive()


if __name__==”__main__”:
   main(sys.argv)
```

Anexo 2 Ficheros PE de MALICIA con igual Imphash y distinta familia

FICHERO 1		FICHERO 2		ATRIBUTO COMPARTIDO
FILE_ID	FAMILY	FILE_ID	FAMILY	IMPHASH
484	securityshield	517	winwebsec	26b4d9c606e432b1165a17264e2a9d92
484	securityshield	518	winwebsec	26b4d9c606e432b1165a17264e2a9d92
484	securityshield	519	winwebsec	26b4d9c606e432b1165a17264e2a9d92
525	securityshield	556	winwebsec	8b53af7395cccfc07e9646fce6a478f7
525	securityshield	557	winwebsec	8b53af7395cccfc07e9646fce6a478f7
525	securityshield	558	winwebsec	8b53af7395cccfc07e9646fce6a478f7
525	securityshield	562	winwebsec	8b53af7395cccfc07e9646fce6a478f7
525	securityshield	564	winwebsec	8b53af7395cccfc07e9646fce6a478f7
525	securityshield	565	winwebsec	8b53af7395cccfc07e9646fce6a478f7
525	securityshield	566	winwebsec	8b53af7395cccfc07e9646fce6a478f7
532	securityshield	556	winwebsec	8b53af7395cccfc07e9646fce6a478f7
532	securityshield	557	winwebsec	8b53af7395cccfc07e9646fce6a478f7
532	securityshield	558	winwebsec	8b53af7395cccfc07e9646fce6a478f7
532	securityshield	562	winwebsec	8b53af7395cccfc07e9646fce6a478f7
532	securityshield	564	winwebsec	8b53af7395cccfc07e9646fce6a478f7
532	securityshield	565	winwebsec	8b53af7395cccfc07e9646fce6a478f7
532	securityshield	566	winwebsec	8b53af7395cccfc07e9646fce6a478f7
534	securityshield	556	winwebsec	8b53af7395cccfc07e9646fce6a478f7
534	securityshield	557	winwebsec	8b53af7395cccfc07e9646fce6a478f7
534	securityshield	558	winwebsec	8b53af7395cccfc07e9646fce6a478f7
534	securityshield	562	winwebsec	8b53af7395cccfc07e9646fce6a478f7
534	securityshield	564	winwebsec	8b53af7395cccfc07e9646fce6a478f7
534	securityshield	565	winwebsec	8b53af7395cccfc07e9646fce6a478f7
534	securityshield	566	winwebsec	8b53af7395cccfc07e9646fce6a478f7
536	securityshield	556	winwebsec	8b53af7395cccfc07e9646fce6a478f7
536	securityshield	557	winwebsec	8b53af7395cccfc07e9646fce6a478f7
536	securityshield	558	winwebsec	8b53af7395cccfc07e9646fce6a478f7
536	securityshield	562	winwebsec	8b53af7395cccfc07e9646fce6a478f7

536	securityshield	564	winwebsec	8b53af7395cccfc07e9646fce6a478f7
536	securityshield	565	winwebsec	8b53af7395cccfc07e9646fce6a478f7
536	securityshield	566	winwebsec	8b53af7395cccfc07e9646fce6a478f7
543	securityshield	556	winwebsec	8b53af7395cccfc07e9646fce6a478f7
543	securityshield	557	winwebsec	8b53af7395cccfc07e9646fce6a478f7
543	securityshield	558	winwebsec	8b53af7395cccfc07e9646fce6a478f7
543	securityshield	562	winwebsec	8b53af7395cccfc07e9646fce6a478f7
543	securityshield	564	winwebsec	8b53af7395cccfc07e9646fce6a478f7
543	securityshield	565	winwebsec	8b53af7395cccfc07e9646fce6a478f7
543	securityshield	566	winwebsec	8b53af7395cccfc07e9646fce6a478f7
548	securityshield	556	winwebsec	8b53af7395cccfc07e9646fce6a478f7
548	securityshield	557	winwebsec	8b53af7395cccfc07e9646fce6a478f7
548	securityshield	558	winwebsec	8b53af7395cccfc07e9646fce6a478f7
548	securityshield	562	winwebsec	8b53af7395cccfc07e9646fce6a478f7
548	securityshield	564	winwebsec	8b53af7395cccfc07e9646fce6a478f7
548	securityshield	565	winwebsec	8b53af7395cccfc07e9646fce6a478f7
548	securityshield	566	winwebsec	8b53af7395cccfc07e9646fce6a478f7
550	securityshield	556	winwebsec	8b53af7395cccfc07e9646fce6a478f7
550	securityshield	557	winwebsec	8b53af7395cccfc07e9646fce6a478f7
550	securityshield	558	winwebsec	8b53af7395cccfc07e9646fce6a478f7
550	securityshield	562	winwebsec	8b53af7395cccfc07e9646fce6a478f7
550	securityshield	564	winwebsec	8b53af7395cccfc07e9646fce6a478f7
550	securityshield	565	winwebsec	8b53af7395cccfc07e9646fce6a478f7
550	securityshield	566	winwebsec	8b53af7395cccfc07e9646fce6a478f7
555	securityshield	556	winwebsec	8b53af7395cccfc07e9646fce6a478f7
555	securityshield	557	winwebsec	8b53af7395cccfc07e9646fce6a478f7
555	securityshield	558	winwebsec	8b53af7395cccfc07e9646fce6a478f7
555	securityshield	562	winwebsec	8b53af7395cccfc07e9646fce6a478f7
555	securityshield	564	winwebsec	8b53af7395cccfc07e9646fce6a478f7
555	securityshield	565	winwebsec	8b53af7395cccfc07e9646fce6a478f7
555	securityshield	566	winwebsec	8b53af7395cccfc07e9646fce6a478f7
556	winwebsec	560	securityshield	8b53af7395cccfc07e9646fce6a478f7
556	winwebsec	563	securityshield	8b53af7395cccfc07e9646fce6a478f7

556	winwebsec	572	securityshield	8b53af7395cccfc07e9646fce6a478f7
556	winwebsec	583	securityshield	8b53af7395cccfc07e9646fce6a478f7
557	winwebsec	560	securityshield	8b53af7395cccfc07e9646fce6a478f7
557	winwebsec	563	securityshield	8b53af7395cccfc07e9646fce6a478f7
557	winwebsec	572	securityshield	8b53af7395cccfc07e9646fce6a478f7
557	winwebsec	583	securityshield	8b53af7395cccfc07e9646fce6a478f7
558	winwebsec	560	securityshield	8b53af7395cccfc07e9646fce6a478f7
558	winwebsec	563	securityshield	8b53af7395cccfc07e9646fce6a478f7
558	winwebsec	572	securityshield	8b53af7395cccfc07e9646fce6a478f7
558	winwebsec	583	securityshield	8b53af7395cccfc07e9646fce6a478f7
560	securityshield	562	winwebsec	8b53af7395cccfc07e9646fce6a478f7
560	securityshield	564	winwebsec	8b53af7395cccfc07e9646fce6a478f7
560	securityshield	565	winwebsec	8b53af7395cccfc07e9646fce6a478f7
560	securityshield	566	winwebsec	8b53af7395cccfc07e9646fce6a478f7
562	winwebsec	563	securityshield	8b53af7395cccfc07e9646fce6a478f7
562	winwebsec	572	securityshield	8b53af7395cccfc07e9646fce6a478f7
562	winwebsec	583	securityshield	8b53af7395cccfc07e9646fce6a478f7
563	securityshield	564	winwebsec	8b53af7395cccfc07e9646fce6a478f7
563	securityshield	565	winwebsec	8b53af7395cccfc07e9646fce6a478f7
563	securityshield	566	winwebsec	8b53af7395cccfc07e9646fce6a478f7
564	winwebsec	572	securityshield	8b53af7395cccfc07e9646fce6a478f7
564	winwebsec	583	securityshield	8b53af7395cccfc07e9646fce6a478f7
565	winwebsec	572	securityshield	8b53af7395cccfc07e9646fce6a478f7
565	winwebsec	583	securityshield	8b53af7395cccfc07e9646fce6a478f7
566	winwebsec	572	securityshield	8b53af7395cccfc07e9646fce6a478f7
566	winwebsec	583	securityshield	8b53af7395cccfc07e9646fce6a478f7
688	securityshield	689	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	690	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	691	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	692	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	693	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	697	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	698	winwebsec	b5cb147a043775fa995dbb94670cd866

688	securityshield	699	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	707	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	708	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	709	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	713	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	714	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	715	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	718	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	719	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	720	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	724	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	725	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	726	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	729	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	730	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	731	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	735	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	736	winwebsec	b5cb147a043775fa995dbb94670cd866
688	securityshield	737	winwebsec	b5cb147a043775fa995dbb94670cd866
689	winwebsec	710	securityshield	b5cb147a043775fa995dbb94670cd866
689	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
689	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
690	winwebsec	710	securityshield	b5cb147a043775fa995dbb94670cd866
690	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
690	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
691	winwebsec	710	securityshield	b5cb147a043775fa995dbb94670cd866
691	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
691	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
692	winwebsec	710	securityshield	b5cb147a043775fa995dbb94670cd866
692	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
692	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
693	winwebsec	710	securityshield	b5cb147a043775fa995dbb94670cd866
693	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866

693	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
697	winwebsec	710	securityshield	b5cb147a043775fa995dbb94670cd866
697	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
697	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
698	winwebsec	710	securityshield	b5cb147a043775fa995dbb94670cd866
698	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
698	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
699	winwebsec	710	securityshield	b5cb147a043775fa995dbb94670cd866
699	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
699	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
707	winwebsec	710	securityshield	b5cb147a043775fa995dbb94670cd866
707	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
707	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
708	winwebsec	710	securityshield	b5cb147a043775fa995dbb94670cd866
708	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
708	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
709	winwebsec	710	securityshield	b5cb147a043775fa995dbb94670cd866
709	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
709	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
710	securityshield	713	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	714	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	715	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	718	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	719	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	720	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	724	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	725	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	726	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	729	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	730	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	731	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	735	winwebsec	b5cb147a043775fa995dbb94670cd866
710	securityshield	736	winwebsec	b5cb147a043775fa995dbb94670cd866

710	securityshield	737	winwebsec	b5cb147a043775fa995dbb94670cd866
713	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
713	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
714	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
714	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
715	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
715	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
718	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
718	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
719	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
719	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
720	winwebsec	721	securityshield	b5cb147a043775fa995dbb94670cd866
720	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
721	securityshield	724	winwebsec	b5cb147a043775fa995dbb94670cd866
721	securityshield	725	winwebsec	b5cb147a043775fa995dbb94670cd866
721	securityshield	726	winwebsec	b5cb147a043775fa995dbb94670cd866
721	securityshield	729	winwebsec	b5cb147a043775fa995dbb94670cd866
721	securityshield	730	winwebsec	b5cb147a043775fa995dbb94670cd866
721	securityshield	731	winwebsec	b5cb147a043775fa995dbb94670cd866
721	securityshield	735	winwebsec	b5cb147a043775fa995dbb94670cd866
721	securityshield	736	winwebsec	b5cb147a043775fa995dbb94670cd866
721	securityshield	737	winwebsec	b5cb147a043775fa995dbb94670cd866
724	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
725	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
726	winwebsec	727	securityshield	b5cb147a043775fa995dbb94670cd866
727	securityshield	729	winwebsec	b5cb147a043775fa995dbb94670cd866
727	securityshield	730	winwebsec	b5cb147a043775fa995dbb94670cd866
727	securityshield	731	winwebsec	b5cb147a043775fa995dbb94670cd866
727	securityshield	735	winwebsec	b5cb147a043775fa995dbb94670cd866
727	securityshield	736	winwebsec	b5cb147a043775fa995dbb94670cd866
727	securityshield	737	winwebsec	b5cb147a043775fa995dbb94670cd866
869	winwebsec	880	securityshield	c66f0821668aebc35f5e6a38718761f4
869	winwebsec	884	securityshield	c66f0821668aebc35f5e6a38718761f4

870	winwebsec	880	securityshield	c66f0821668aebc35f5e6a38718761f4
870	winwebsec	884	securityshield	c66f0821668aebc35f5e6a38718761f4
871	winwebsec	880	securityshield	c66f0821668aebc35f5e6a38718761f4
871	winwebsec	884	securityshield	c66f0821668aebc35f5e6a38718761f4
872	winwebsec	880	securityshield	c66f0821668aebc35f5e6a38718761f4
872	winwebsec	884	securityshield	c66f0821668aebc35f5e6a38718761f4
873	winwebsec	880	securityshield	c66f0821668aebc35f5e6a38718761f4
873	winwebsec	884	securityshield	c66f0821668aebc35f5e6a38718761f4
1152	winwebsec	1226	securityshield	fab919a5ff86dd64eecd7032969e9193
1153	winwebsec	1226	securityshield	fab919a5ff86dd64eecd7032969e9193
1154	winwebsec	1226	securityshield	fab919a5ff86dd64eecd7032969e9193
1160	winwebsec	1212	securityshield	c1ada0e588c7b2b7f03dbd1d000470e6
1212	securityshield	1231	winwebsec	c1ada0e588c7b2b7f03dbd1d000470e6
1212	securityshield	1232	winwebsec	c1ada0e588c7b2b7f03dbd1d000470e6
1212	securityshield	1233	winwebsec	c1ada0e588c7b2b7f03dbd1d000470e6
1224	winwebsec	1226	securityshield	fab919a5ff86dd64eecd7032969e9193
1225	winwebsec	1226	securityshield	fab919a5ff86dd64eecd7032969e9193
1317	securityshield	1325	winwebsec	0feb01bcec3d86640f7b4a49726d6e06
1317	securityshield	1326	winwebsec	0feb01bcec3d86640f7b4a49726d6e06
1317	securityshield	1327	winwebsec	0feb01bcec3d86640f7b4a49726d6e06
1317	securityshield	1360	winwebsec	0feb01bcec3d86640f7b4a49726d6e06
1317	securityshield	1369	winwebsec	0feb01bcec3d86640f7b4a49726d6e06
1317	securityshield	1370	winwebsec	0feb01bcec3d86640f7b4a49726d6e06
1317	securityshield	1371	winwebsec	0feb01bcec3d86640f7b4a49726d6e06
1319	winwebsec	1393	securityshield	e2d95e21277712bf3ff8745ae131b707
1320	winwebsec	1393	securityshield	e2d95e21277712bf3ff8745ae131b707
1321	winwebsec	1393	securityshield	e2d95e21277712bf3ff8745ae131b707
1356	winwebsec	1393	securityshield	e2d95e21277712bf3ff8745ae131b707
1357	winwebsec	1393	securityshield	e2d95e21277712bf3ff8745ae131b707
1508	winwebsec	1656	securityshield	4efa911c78a20c1d8b683b879b7e0a00
1509	winwebsec	1656	securityshield	4efa911c78a20c1d8b683b879b7e0a00
1510	winwebsec	1656	securityshield	4efa911c78a20c1d8b683b879b7e0a00
9578	zbot	9942	CLUSTER:foreign	888dde8753b35c494db3e8d44f2096e7

12196	cridex	12302	CLUSTER:dzony3777.su	dba6ef29c14cb57a68a9122782240b48
12196	cridex	12303	CLUSTER:91.234.32.10	dba6ef29c14cb57a68a9122782240b48
12228	cridex	12302	CLUSTER:dzony3777.su	dba6ef29c14cb57a68a9122782240b48
12228	cridex	12303	CLUSTER:91.234.32.10	dba6ef29c14cb57a68a9122782240b48
12302	CLUSTER:dzony3777.su	12303	CLUSTER:91.234.32.10	dba6ef29c14cb57a68a9122782240b48
12501	zbot	12515	cridex	e98e78a7162fdab47c07b6e6695f50cf
12742	cridex	12743	harebot	a4b4cd2bde1236dbed4470e806890583